REPUBLICA BOLIVARIANA DE VENEZUELA

INSTITUTO UNIVERSITARIO DE TECNOLOGÍA INDUSTRIAL

“RODOLFO LOERO ARISMENDI”

I.U.T.I.R.L.A

EXTENSIÓN MATURÍN



JAVA

Profesora: Bachiller:

Milagro Rodríguez Humberto Ávila

Maturín, Octubre de 2015

INDICE

1.- INTRODUCCIÓN……………………………………………………………………….1

2.- HISTORIA DE JAVA…………………………………………………………………...2

3.- CARACTERÍSTICAS DE JAVA……………………………………………………….6

4.- TIPOS DE DATOS (VARIABLES) EN JAVA…………………………………………9

5.- HERRAMIENTAS DE DESARROLLO………………………………………………14

6.- VENTAJAS Y DESVENTAJAS DE JAVA…………………………………………...16

7.- ELEMENTOS DEL LENGUAJE JAVA………………………………………………19

8.- OPERADORES LÓGICOS PRINCIPALES EN JAVA……………………………….29

#### 9.- ESTRUCTURA BÁSICA DE JAVA……………………………………………..........32

10.- CONCLUSIÓN……………………………………………………………………......33

11.- BIBLIOGRAFIA……………………………………………………………………...34

## INTRODUCCIÓN

Java es un [lenguaje de programación](http://www.monografias.com/Computacion/Programacion/) orientado a objetos, desarrollado por Sun Microsystems a [principios](http://www.monografias.com/trabajos6/etic/etic.shtml) de los años 90. [El lenguaje](http://www.monografias.com/trabajos16/desarrollo-del-lenguaje/desarrollo-del-lenguaje.shtml) en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un [modelo](http://www.monografias.com/trabajos/adolmodin/adolmodin.shtml) de objetos más simple y elimina [herramientas](http://www.monografias.com/trabajos11/contrest/contrest.shtml) de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o [memoria](http://www.monografias.com/trabajos13/memor/memor.shtml).   
Las aplicaciones [Java](http://www.monografias.com/trabajos16/java/java.shtml) están típicamente compiladas en un bytecode, aunque la compilación en [código](http://www.monografias.com/trabajos12/eticaplic/eticaplic.shtml) máquina nativo también es posible. En el [tiempo](http://www.monografias.com/trabajos901/evolucion-historica-concepciones-tiempo/evolucion-historica-concepciones-tiempo.shtml) de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por [hardware](http://www.monografias.com/Computacion/Hardware/) del bytecode por un [procesador](http://www.monografias.com/trabajos5/sisope/sisope.shtml) Java también es posible.   
La implementación original y de referencia del compilador, la máquina virtual y las [bibliotecas](http://www.monografias.com/trabajos10/ponency/ponency.shtml) de clases de Java fueron desarrollados por Sun Microsystems en 1995.

**La importancia de Java radica en la posibilidad de crear o diseñar software y poder ser ejecutado en diferentes plataformas donde se requiere, sin la necesidad de estar modificándolo. Además de esto, es una herramienta diseñada con muchos elementos parecidos a C++ pero en una versión mejorada, más sencilla de manejar, con menos complejidad y con barreras de seguridad en su lenguaje y ejecución.**

HISTORIA DE JAVA

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 el año [1991](https://es.wikipedia.org/wiki/1991). El equipo (Green Team), compuesto por trece personas y dirigido por [James Gosling](https://es.wikipedia.org/wiki/James_Gosling), trabajó durante 18 meses en Sand Hill Road en Menlo Park en su desarrollo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a denominarseGreen tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas y finalmente se renombró a 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 ("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 4 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.

Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a [C++](https://es.wikipedia.org/wiki/C%2B%2B). Entre junio y julio de 1994, tras una sesión maratoniana de tres días entre John Gaga, 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](https://es.wikipedia.org/wiki/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](https://es.wikipedia.org/wiki/HotJava" \o "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 encargase 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](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java)) 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](https://es.wikipedia.org/wiki/Applet" \o "Applet) Java incrustadas en las páginas web.

Java ha experimentado numerosos cambios desde la versión primigenia, [JDK](https://es.wikipedia.org/wiki/JDK) 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar.

Desde [J2SE](https://es.wikipedia.org/wiki/Java_SE) 1.4, la evolución del lenguaje ha sido regulada por el JCP ([Java Community Process](https://es.wikipedia.org/wiki/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. Los cambios en los JLS son gestionados en [JSR 901](http://www.jcp.org/en/jsr/detail?id=63).

* JDK 1.0 ([23 de enero](https://es.wikipedia.org/wiki/23_de_enero) de [1996](https://es.wikipedia.org/wiki/1996)) — Primer lanzamiento: [comunicado de prensa](http://www.google.com/)
* JDK 1.1 ([19 de febrero](https://es.wikipedia.org/wiki/19_de_febrero) de [1997](https://es.wikipedia.org/wiki/1997)) — Principales adiciones incluidas: [comunicado de prensa](http://www.sun.com/smi/Press/sunflash/1997-02/sunflash.970219.0001.html)
  + una reestructuración intensiva del modelo de eventos AWT (Abstract Windowing Toolkit)
  + clases internas (inner classes)
  + [JavaBeans](https://es.wikipedia.org/wiki/JavaBeans)
  + [JDBC](https://es.wikipedia.org/wiki/JDBC) (Java Database Connectivity), para la integración de bases de datos
  + [RMI](https://es.wikipedia.org/wiki/Java_Remote_Method_Invocation) (Remote Method Invocation)
* J2SE 1.2 ([8 de diciembre](https://es.wikipedia.org/wiki/8_de_diciembre) de [1998](https://es.wikipedia.org/wiki/1998)) — Nombre clave Playground. Esta y las siguientes versiones fueron recogidas bajo la denominación Java 2 y el nombre "J2SE" (Java 2 Platform, Standard Edition), reemplazó a JDK para distinguir la plataforma base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition). Otras mejoras añadidas incluían: [comunicado de prensa](http://www.sun.com/smi/Press/sunflash/1998-12/sunflash.981208.9.html)
  + la palabra reservada (keyword) [strictfp](https://es.wikipedia.org/wiki/Strictfp" \o "Strictfp)
  + [reflexión en la programación](https://es.wikipedia.org/wiki/Reflexi%C3%B3n_(inform%C3%A1tica))
  + la API gráfica ( [Swing](https://es.wikipedia.org/wiki/Swing_(biblioteca_gr%C3%A1fica))) fue integrada en las clases básicas
  + la máquina virtual (JVM) de Sun fue equipada con un [compilador](https://es.wikipedia.org/wiki/Compilador) [JIT](https://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n) (Just in Time) por primera vez
  + [Java Plug-in](https://es.wikipedia.org/w/index.php?title=Java_Plug-in&action=edit&redlink=1)
  + [Java IDL](https://es.wikipedia.org/wiki/Java_IDL), una implementación de IDL (Lenguaje de Descripción de [Interfaz](https://es.wikipedia.org/wiki/Interfaz_(Java))) para la interoperabilidad con [CORBA](https://es.wikipedia.org/wiki/CORBA)
  + Colecciones ([Collections](https://es.wikipedia.org/w/index.php?title=Collection_class&action=edit&redlink=1" \o "Collection class (aún no redactado)))
* J2SE 1.3 ([8 de mayo](https://es.wikipedia.org/wiki/8_de_mayo) de [2000](https://es.wikipedia.org/wiki/2000)) — Nombre clave Kestrel. Los cambios más notables fueron:[comunicado de prensa](http://www.sun.com/smi/Press/sunflash/2000-05/sunflash.20000508.3.html) [lista completa de cambios](http://java.sun.com/j2se/1.3/docs/relnotes/features.html)
  + la inclusión de la máquina virtual de [HotSpot](https://es.wikipedia.org/w/index.php?title=HotSpot&action=edit&redlink=1" \o "HotSpot (aún no redactado)) JVM (la JVM de HotSpot fue lanzada inicialmente en abril de 1999, para la JVM de J2SE 1.2)
  + [RMI](https://es.wikipedia.org/wiki/Java_Remote_Method_Invocation) fue cambiado para que se basara en [CORBA](https://es.wikipedia.org/wiki/CORBA)
  + [JavaSound](https://es.wikipedia.org/wiki/JavaSound)
  + se incluyó el [Java Naming and Directory Interface](https://es.wikipedia.org/wiki/JNDI) (JNDI) en el paquete de bibliotecas principales (anteriormente disponible como una extensión)
  + [Java Platform Debugger Architecture](https://es.wikipedia.org/wiki/Java_Platform_Debugger_Architecture) (JPDA)
* J2SE 1.4 ([6 de febrero](https://es.wikipedia.org/wiki/6_de_febrero) de [2002](https://es.wikipedia.org/wiki/2002)) — Nombre Clave Merlin. Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como [JSR 59](http://www.jcp.org/en/jsr/detail?id=59). Los cambios más notables fueron: [comunicado de prensa](http://www.sun.com/smi/Press/sunflash/2002-02/sunflash.20020206.5.html)[lista completa de cambios](http://java.sun.com/j2se/1.4.2/docs/relnotes/features.html)
  + Palabra reservada [assert](https://es.wikipedia.org/w/index.php?title=Assertion_(computing)&action=edit&redlink=1" \o "Assertion (computing) (aún no redactado)) (Especificado en [JSR 41](http://www.jcp.org/en/jsr/detail?id=41).)
  + [Expresiones regulares](https://es.wikipedia.org/wiki/Expresiones_regulares) modeladas al estilo de las expresiones regulares [Perl](https://es.wikipedia.org/wiki/Perl_programming_language)
  + [Encadenación de excepciones](https://es.wikipedia.org/w/index.php?title=Encadenaci%C3%B3n_de_excepciones&action=edit&redlink=1) Permite a una excepción encapsular la excepción de bajo nivel original.
  + non-blocking NIO ([New Input/Output](https://es.wikipedia.org/w/index.php?title=New_I/O&action=edit&redlink=1)) (Especificado en [JSR 51](http://www.jcp.org/en/jsr/detail?id=51).)
  + Logging API (Specified in [JSR 47](http://www.jcp.org/en/jsr/detail?id=47).)
  + API I/O para la lectura y escritura de imágenes en formatos como [JPEG](https://es.wikipedia.org/wiki/JPEG) o [PNG](https://es.wikipedia.org/wiki/PNG)
  + Parser [XML](https://es.wikipedia.org/wiki/XML) integrado y procesador [XSLT](https://es.wikipedia.org/wiki/XSLT) ([JAXP](https://es.wikipedia.org/w/index.php?title=JAXP&action=edit&redlink=1)) (Especificado en [JSR 5](http://www.jcp.org/en/jsr/detail?id=5) y [JSR 63](http://www.jcp.org/en/jsr/detail?id=63).)
  + Seguridad integrada y extensiones criptográficas (JCE, [JSSE](https://es.wikipedia.org/wiki/JSSE), [JAAS](https://es.wikipedia.org/wiki/JAAS))
  + [Java Web Start](https://es.wikipedia.org/wiki/Java_Web_Start) incluido (El primer lanzamiento ocurrió en marzo de 2001 para J2SE 1.3) (Especificado en [JSR 56](http://www.jcp.org/en/jsr/detail?id=56).)
* J2SE 5.0 ([30 de septiembre](https://es.wikipedia.org/wiki/30_de_septiembre) de [2004](https://es.wikipedia.org/wiki/2004)) — Nombre clave: Tiger. (Originalmente numerado 1.5, esta notación aún es usada internamente.[[2]](http://java.sun.com/j2se/1.5.0/docs/relnotes/version-5.0.html)) Desarrollado bajo[JSR 176](http://www.jcp.org/en/jsr/detail?id=176), Tiger añadió un número significativo de nuevas características [comunicado de prensa](http://www.sun.com/smi/Press/sunflash/2004-09/sunflash.20040930.1.html)
  + [Plantillas (genéricos)](https://es.wikipedia.org/wiki/Programaci%C3%B3n_gen%C3%A9rica) — provee [conversión de tipos (type safety)](https://es.wikipedia.org/w/index.php?title=Conversi%C3%B3n_de_tipos_(type_safety)&action=edit&redlink=1) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de[conversión de tipos (type casting)](https://es.wikipedia.org/wiki/Conversi%C3%B3n_de_tipos). (Especificado por [JSR 14](http://www.jcp.org/en/jsr/detail?id=14).)
  + [Metadatos](https://es.wikipedia.org/wiki/Metadatos) — también llamados [anotaciones](https://es.wikipedia.org/w/index.php?title=Anotacione&action=edit&redlink=1), permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos. (Especificado por [JSR 175](http://www.jcp.org/en/jsr/detail?id=175).)
  + [Autoboxing](https://es.wikipedia.org/w/index.php?title=Autoboxing&action=edit&redlink=1)/unboxing — Conversiones automáticas entre [tipos primitivos](https://es.wikipedia.org/wiki/Tipo_primitivo) (Como los int) y [clases de envoltura primitivas](https://es.wikipedia.org/w/index.php?title=Clases_de_envoltura_primitivas&action=edit&redlink=1) (Como [Integer](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Integer.html)). (Especificado por [JSR 201](http://www.jcp.org/en/jsr/detail?id=201).)
  + [Enumeraciones](https://es.wikipedia.org/wiki/Enumeraci%C3%B3n) — la palabra reservada enum crea una [typesafe](https://es.wikipedia.org/w/index.php?title=Type_safety&action=edit&redlink=1" \o "Type safety (aún no redactado)), lista ordenada de valores (como Dia.LUNES, Dia.MARTES, etc.). Anteriormente, esto solo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (enum pattern). (Especificado por [JSR 201](http://www.jcp.org/en/jsr/detail?id=201).)
  + Varargs (número de argumentos variable) — El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos (e.g.void drawtext(String... lines)). En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un array para pasarlos al método.
  + [Bucle](https://es.wikipedia.org/wiki/Bucle_(programaci%C3%B3n)) for mejorado — La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente [Iterable](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Iterable.html), como la clase estándar [Collection](http://java.sun.com/j2se/1.5.0/docs/api/java/util/Collection.html), de la siguiente forma:

void displayWidgets (Iterable<Widget> widgets) {

for (Widget w : widgets) {

w.display();

}

}

Este ejemplo itera sobre el objeto Iterable widgets, asignando, en orden, cada uno de los elementos a la variable w, y llamando al método display() de cada uno de ellos. (Especificado por JSR 201.)

* Java SE 6 ([11 de diciembre](https://es.wikipedia.org/wiki/11_de_diciembre) de [2006](https://es.wikipedia.org/wiki/2006)) — Nombre clave [Mustang](https://mustang.dev.java.net/). Estuvo en desarrollo bajo la [JSR 270](http://www.jcp.org/en/jsr/detail?id=270). En esta versión, Sun cambió el nombre "J2SE" porJava SE y eliminó el ".0" del número de versión.. Está disponible en <http://java.sun.com/javase/6/>. Los cambios más importantes introducidos en esta versión son:
  + Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript.
  + Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java.
  + Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP.
  + Mejoras en la interfaz gráfica y en el rendimiento.
* Java SE 7 — Nombre clave Dolphin. En el año [2006](https://es.wikipedia.org/wiki/2006) aún se encontraba en las primeras etapas de planificación. Su lanzamiento fue en julio de 2011.
  + Soporte para XML dentro del propio lenguaje.
  + Un nuevo concepto de superpaquete.
  + Soporte para [closures](https://es.wikipedia.org/wiki/Clausura_(inform%C3%A1tica)" \o "Clausura (informática)).
  + Introducción de anotaciones estándar para detectar fallos en el software.
* No oficiales:
  + NIO2.
  + Java Module System.
  + Java Kernel.
  + Nueva API para el manejo de Días y Fechas, la cual reemplazara las antiguas clases Date y Calendar.
  + Posibilidad de operar con clases BigDecimal usando operandos.
* Java SE 8 — lanzada en marzo de 2014. Cabe destacar:
  + Incorpora de forma completa la librería JavaFX.
  + Diferentes mejoras en seguridad.
  + Diferentes mejoras en concurrencia.
  + Añade funcionalidad para programación funcional mediante expresiones Lambda.
  + Mejora la integración de JavaScript.
  + Nuevas API para manejo de fechas y tiempo (date - time).

En el [2005](https://es.wikipedia.org/wiki/2005) se calcula en 4,5 millones el número de desarrolladores y 2.500 millones de dispositivos habilitados con tecnología Java.

CARACTERÍSTICAS DE JAVA

En [http://java.sun.com](http://java.sun.com/) se puede leer que Java es:

"Un lenguaje simple. Orientado al objeto, distribuido, interpretado, sólido, seguro, de arquitectura neutral, portable, de lato desempeño, de multihilos y dinámico"

Simple  
Basado en el lenguaje C++ pero donde se eliminan muchas de las características OOP que se utilizan esporádicamente y que creaban frecuentes problemas a los programadores. Esta eliminación de causas de error y problemas de mantenimiento facilita y reduce el coste del desarrollo de software.

* + Java no da soporte a struct, union y pointer
  + Java no ofrece typedef ni #define
  + No permite la sobrecarga de operadores.
  + No ofrece herencia múltiple.
  + Maneja los comandos en línea de diferente manera que C++
  + Java tienen una clase String, que permite un mejor manejo que los arrays de terminación nula del C y C++.
  + Java tiene un sistema automático de asignación y liberación de memoria (recolector de basura) que mejora mucho los sistemas del C++
  + Orientado al objeto  
    Java da buen soporte a las técnicas de desarrollo OOP y en resumen a la reutilización de componentes de software.
  + Distribuido  
    Java se ha diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP. El código Java se puede manipular a través de recursos URL con la misma facilidad que C y C++ utilizan recursos locales (archivos).
  + Interpretado  
    El compilador Java traduce cada fichero fuente de clases a código de bytes (Bytecode), que puede ser interpretado por todas las máquinas que den soporte a un visualizador de que funcione con Java. Este Bytecode no es especifico de una máquina determinada, por lo que no se compila y enlaza como en el ciclo clásico, sino que se interpreta.
  + Sólido  
    El código Java no se quiebra fácilmente ante errores de programación. Así el relaje que existe en la declaración y manejo de tipos en C y C++ se torna en restricciones en Java, donde no es posible la conversión forzada (cast) de enteros en punteros y no ofrece soporte a los punteros que permitan saltarse reglas de manejo de tipos. Así en Java no es posible escribir en áreas arbitrarias de memoria ni realizar operaciones que corrompan el código. En resumen se eliminan muchas de las posibilidades de "trucos" que ofrecía el C y C++.
  + Seguro  
    Como Java suele funcionar en ambiente de redes el tema de seguridad debe interesar en sobremanera. Las mismas características antes descritas que evitan la corrupción de código evitan su manipulación. Actualmente se esta trabajando en encriptar el código.
  + Arquitectura neutral  
    El compilador crea códigos de byte (Bytecode) que se envía al visualizador solicitado y se interpreta en la máquina que posee un interprete de Java o dispone de un visualizador que funciona con Java.
  + Portable  
    Al ser de arquitectura neutral es altamente portable, pero esta característica puede verse de otra manera: Los tipos estándares (int, float ...) están igualmente implementados en todas las máquinas por lo que las operaciones aritméticas funcionaran igual en todas las máquinas.
  + Alto desempeño  
    al ser código interpretado, la ejecución no es tan rápida como el código compilado para una plataforma particular. El compilador Java suele ofrecer la posibilidad de compilar Bytecode en código máquina de determinadas plataformas, y según Sun este código resultar de una eficacia similar a compilaciones de C y C++.
  + Multihilos  
    Java puede aplicarse a la realización de aplicaciones en las que ocurra más de una cosa a la vez. Java, apoyándose en un sistema de gestión de eventos basado en el paradigma de condición y monitores C.A.R. permite apoyar la conducta en tiempo real e interactiva en programas
  + Dinámico  
    al contrario que C++ que exige se compile de nuevo la aplicación al cambiar una clase madre Java utiliza un sistema de interfaces que permite aligerar esta dependencia. Como resultado, los programas Java pueden permitir nuevos métodos y variables en un objeto de biblioteca sin afectar a los objetos dependientes.

TIPOS DE DATOS (VARIABLES) EN JAVA.

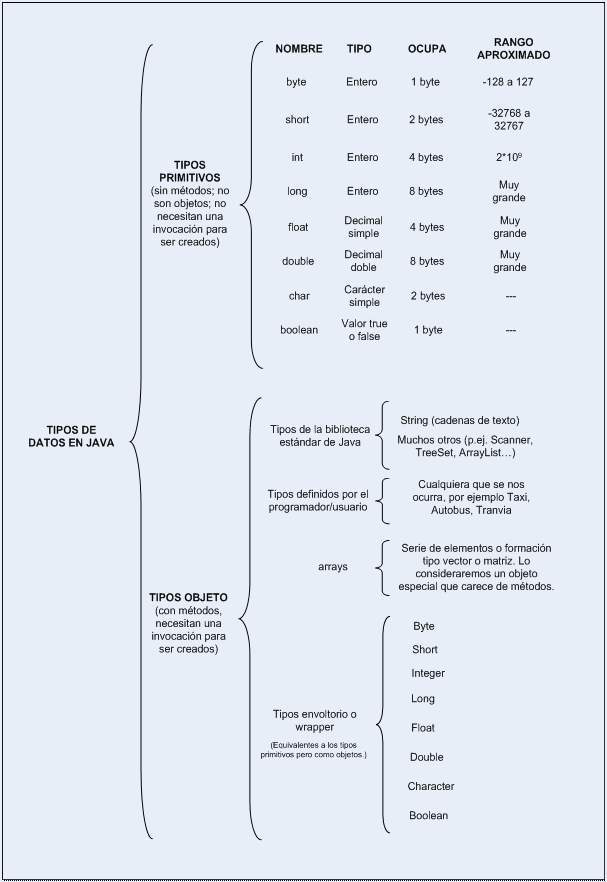
Los primeros lenguajes de programación no usaban objetos, solo variables. Una variable podríamos decir que es un espacio de la memoria del ordenador a la que asignamos un contenido que puede ser un valor numérico (sólo números, con su valor de cálculo) o de tipo carácter o cadena de caracteres (valor alfanumérico que constará sólo de texto o de texto mezclado con números).

Como ejemplo podemos definir una variable a que contenga 32 y esto lo escribimos como a= 32. Posteriormente podemos cambiar el valor de a y hacer a = 78. O hacer “a” equivalente al valor de otra variable “b” así: a = b.

Dado que antes hemos dicho que un objeto también ocupa un espacio de memoria: ¿en qué se parecen y en qué se diferencia un objeto de una variable? Consideraremos que las variables son entidades elementales: un número, un carácter, un valor verdadero o falso… mientras que los objetos son entidades complejas que pueden estar formadas por la agrupación de muchas variables y métodos. Pero ambas cosas ocupan lo mismo: un espacio de memoria (que puede ser más o menos grande).

En los programas en Java puede ser necesario tanto el uso de datos elementales como de datos complejos. Por eso en Java se usa el término “Tipos de datos” para englobar a cualquier cosa que ocupa un espacio de memoria y que puede ir tomando distintos valores o características durante la ejecución del programa. Es decir, en vez de hablar de tipos de variables o de tipos de objetos, hablaremos simplemente de tipos de datos. Sin embargo, a veces “coloquialmente” no se utiliza la terminología de forma estricta: puedes encontrarte textos o páginas web donde se habla de una variable en alusión a un objeto.

En Java diferenciamos dos tipos de datos: por un lado, los tipos primitivos, que se corresponden con los tipos de variables en lenguajes como C y que son los datos elementales que hemos citado. Por otro lado, los tipos objeto (que normalmente incluyen métodos).Veamos los tipos de datos en Java sobre un esquema de síntesis:



Esquema de síntesis de tipos de datos en Java

Este esquema no es necesario aprendérselo de memoria en todos sus detalles, aunque sí lo iremos memorizando poco a poco a medida que lo utilicemos, por lo menos hasta tener en nuestra cabeza los nombres de todos los tipos primitivos y envoltorio y sus características (si son objetos o no y su rango aproximado). A continuación mostramos el mismo esquema en formato de tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TIPOS DE  DATOS  EN JAVA |  | NOMBRE | TIPO | OCUPA | RANGO  APROXIMADO |
| TIPOS  PRIMITIVOS  (sin métodos; no  son objetos; no  necesitan una  invocación para  ser creados) | byte | Entero | 1 byte | -128 a 127 |
| short | Entero | 2 bytes | -32768 a 32767 |
| int | Entero | 4 bytes | 2\*109 |
| long | Entero | 8 bytes | Muy grande |
| float | Decimal simple | 4 bytes | Muy grande |
| double | Decimal doble | 8 bytes | Muy grande |
| char | Carácter simple | 2 bytes | --- |
| boolean | Valor true o false | 1 byte | --- |
|  |  |  |  |  |
| TIPOS OBJETO  (con métodos,  necesitan una  invocación para  ser creados) | Tipos de la  Biblioteca  estándar de  Java | String (cadenas de texto)  Muchos otros (p.ej. Scanner, TreeSet, ArrayList…) | | |
| Tipos  definidos por  el  programador / usuario | Cualquiera que se nos ocurra, por ejemplo Taxi, Autobus, Tranvia | | |
| arrays | Serie de elementos o formación tipo vector  o matriz. Lo consideraremos un objeto  especial que carece de métodos. | | |
| Tipos  envoltorio o  wrapper  (Equivalentes  a los tipos  primitivos  pero como  objetos.) | Byte | | |
| Short | | |
| Integer | | |
| Long | | |
| Float | | |
| Double | | |
| Character | | |
| Boolean | | |

Vamos a comentar distintas cuestiones:

1. Un objeto es una cosa distinta a un tipo primitivo, aunque “porten” la misma información. Tener siempre presente que los objetos en Java tienen un tipo de tratamiento y los tipos primitivos, otro. Que en un momento dado contengan la misma información no significa en ningún caso que sean lo mismo. Iremos viendo las diferencias entre ambos poco a poco. De momento, recuerda que el tipo primitivo es algo elemental y el objeto algo complejo. Supón una cesta de manzanas en la calle: algo elemental. Supón una cesta de manzanas dentro de una nave espacial (considerando el conjunto nave + cesta): algo complejo. La información que portan puede ser la misma, pero no son lo mismo.

2. ¿Para qué tener esa aparente duplicidad entre tipos primitivos y tipos envoltorio? Esto es una cuestión que atañe a la concepción del lenguaje de programación. Tener en cuenta una cosa un tipo primitivo es un dato elemental y carece de métodos, mientras que un objeto es una entidad compleja y dispone de métodos. Por otro lado, de acuerdo con la especificación de Java, es posible que necesitemos utilizar dentro de un programa un objeto que “porte” como contenido un número entero. Desde el momento en que sea necesario un objeto habremos de pensar en un envoltorio, por ejemplo Integer. Inicialmente nos puede costar un poco distinguir cuándo usar un tipo primitivo y cuándo un envoltorio en situaciones en las que ambos sean válidos. Seguiremos esta regla: usaremos por norma general tipos primitivos. Cuando para la estructura de datos o el proceso a realizar sea necesario un objeto, usaremos un envoltorio.

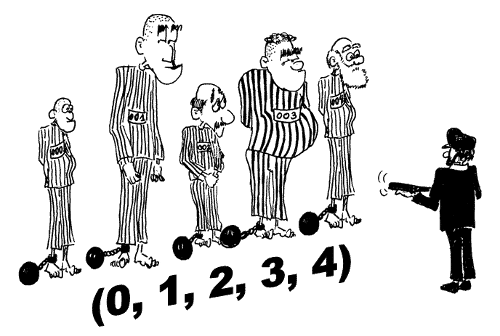
3. Los nombres de tipos primitivos y envoltorio se parecen mucho. En realidad, excepto entre int e Integer y char y Character, la diferencia se limita a que en un caso la inicial es minúscula (por ejemplo double) y en el otro es mayúscula (Double). Esa similitud puede confundirnos inicialmente, pero hemos de tener muy claro qué es cada tipo y cuándo utilizar cada tipo.

4. Una cadena de caracteres es un objeto. El tipo String en Java nos permite crear objetos que contienen texto (palabras, frases, etc.). El texto debe ir siempre entre comillas. Muchas veces se cree erróneamente que el tipo String es un tipo primitivo por analogía con otros lenguajes donde String funciona como una variable elemental. En Java no es así.

5. Hay distintos tipos primitivos enteros. ¿Cuál usar? Por norma general usaremos el tipo int. Para casos en los que el entero pueda ser muy grande usaremos el tipo long. Los tipos byte y short los usaremos cuando tengamos un mayor dominio del lenguaje.

6. ¿Cuántos tipos de la biblioteca estándar de Java hay? Cientos o miles. Es imposible conocerlos todos.

7. ¿Un array es un objeto? Los arrays los consideraremos objetos especiales, los únicos objetos en Java que carecen de métodos.



Concepto de array: serie de elementos, cada uno de los cuales lleva asociado un índice numérico 0, 1, 2, 3, … , n-1

HERRAMIENTAS DE DESARROLLO

Voy a hablar de las herramientas que considero imprescindibles para un desarrollador Java, en concreto para desarrollo J2EE. Puede que alguna se pueda obviar, como por ejemplo, [Maven](http://maven.apache.org/), pero la verdad es que yo no puedo vivir sin él. No incluyo en la lista la maquina virtual, es obvio que sin eso no puedes hacer nada:

* Lo primero es un IDE. Yo uso el [Eclipse](http://www.eclipse.org/), que es el mas extendido, es opensource y es el que mas plugins tiene. Yo aconsejo descargarlo desde este [enlace](http://eclipsesource.com/en/yoxos/). Es una aplicacion web que te permite crearte tu propio paquete de plugins y descargarlo todo en un zip. Es muy útil para trabajar en un proyecto en equipo, ya que puedes distribuir ese zip entre los programadores, y te ahorras tiempo instalando los plugins desde el propio IDE. Otras opciones son [Netbeans](http://www.netbeans.org/) de Oracle (antes de SUN) o el premiado [Intellij IDEA](http://www.jetbrains.com/idea/). Este último es comercial, aunque tienen edición community.
* [Apache Ant](http://ant.apache.org/) o sucedáneos. Es una herramienta para realizar tareas, como puede ser compilar y empaquetar un proyecto, aunque puedes configurar todo tipo de tareas, como desplegar un war en un servidor de aplicaciones, o subirlo por ftp por ejemplo. Utiliza un lenguajeXML para definirlas en un fichero build.xml en la raíz del proyecto. Se integra muy bien en la mayoria de los IDEs. Existen en el mercado alternativas mejores que ant, como [Gradle](http://www.gradle.org/) o [buildr](http://buildr.apache.org/), pero ant esta muy implantado y sigue siendo el que mas se usa.
* [Apache Maven](http://maven.apache.org/). Como ya he comentado, no puedo vivir sin maven. Es una herramienta de gestión de proyectos. Tiene infinidad de plugins. Su caracteristica mas útil, es que te descarga y añade a tu classpath las librerias que usa tu proyecto. Solo tienes que definirlas en un fichero xml. La mayoria de IDEs lo soportan. Para Eclipse, necesitas instalar [m2eclipse](http://m2eclipse.sonatype.org/) de SonaType.
* Un sistema de control de versiones no puede faltar, ni con java ni con cualquier otro lenguaje. El que mas he utilizado es [Subversion](http://subversion.tigris.org/). La mayoría tienen plugins para los IDEs mas conocidos.
* Un editor de texto ligero con coloreado de sintáxis. Yo uso [Notepad++](http://notepad-plus-plus.org/) o [PsPad](http://www.pspad.com/es/) en windows. En linux uso el [GEdit](http://www.gedit.org/).
* [Cygwin](http://www.cygwin.com/). Es como un emulador de linux sobre windows. Siempre lo instalo cuando trabajo en windows porque no me gusta su terminal. Puedes instalar muchos programas de la linea de comandos de linux que puedes elegir en el instalador.
* [Putty](http://www.putty.org/). Sirve para conectarse a un terminal remoto por SSH o telnet desde windows. Si estoy en linux, utilizo el comando ssh desde el terminal.
* Un frontend para la base de datos. Dependera con cual este trabajando. Ahora mismo utilizo[DbVisualizer](http://www.dbvis.com/).
* Navegador con depurador JavaScript. Estoy usando [Chrome](http://www.google.com/chrome/?hl=es), que viene con un depurador muy bueno. Antes usaba [Firefox](http://www.mozilla-europe.org/es/) junto con el plugin [Firebug](http://getfirebug.com/).
* Servidor de aplicaciones. Actualmente uso el [Glassfish](http://www.glassfish.org/).
* Bugtracker. Imprescindible para llevar un seguimiento de los bugs. Actualmente uso [Jira](http://www.atlassian.com/software/jira/), que es el mas extendido. Tambien he usado [Redmine](http://www.redmine.org/), [Trac](http://trac.edgewall.org/) y [Bugzilla](http://www.bugzilla.org/). Uso un plugin para el eclipse, que me permite conectarme al bugtracker, ver los bugs disponibles y gestionarlos sin salirme delIDE.
* Servidor de integración contínua. Uso [Jenkins](http://jenkins-ci.org/) (hasta hace nada llamado Hudson). Sirve para programar los builds de los proyectos. Para configurar los “builds” lo habitual es conectarlo con el servidor de control de versiones. Puedes configurar que ejecute tareas de ant, ‘goals’ de maven por ejemplo o cualquier otro tipo de script.
* [Google](http://www.google.com/). Imprescindible, esto lo usamos todos los programadores, de cualquier lenguaje.
* Una aplicación de notas. Yo uso [Evernote](http://www.evernote.com/). Es un programa para crear notas y capturar páginas web o fragmentos de estas. Yo lo utilizo para almacenar artículos y tutoriales interesantes para futuras referencias, o fragmentos de código, todo clasificado por carpetas y etiquetas. Dispone de aplicación para Windows, Mac, Android e iOS que se sincroniza con la aplicación web .
* Servicio de almacenamiento en la nube. Yo uso [Dropbox](http://www.dropbox.com/). Lo utilizo para guardar documentos relacionados con Java, como manuales de referencia de diversas tecnologías y frameworks. Teniendolos en un sitio, ahorro tiempo a la hora de consultarlos.
* Repositorios de código opensource. [Github](https://github.com/), [google code](http://code.google.com/) y [sourceforge](http://sourceforge.net/). Muchas veces busco código que me pueda ser de utilidad en el proyecto que estoy trabajando.
* Papel y lápiz. A pesar de todos los avances tecnológicos, aún se usa este primitivo modo de escritura.

VENTAJAS Y DESVENTAJAS DE JAVA

VENTAJAS

Primero: No debes volver a escribir el código si quieres ejecutar el programa en otra máquina. Un solo código funciona para todos los browsers compatibles con Java o donde se tenga una Máquina Virtual de Java (Mac's, PC's, Sun's, etc).

Segundo: Java es un [lenguaje de programación](http://www.monografias.com/Computacion/Programacion/) orientado a objetos, y tiene todos los beneficios que ofrece esta [metodología](http://www.monografias.com/trabajos11/metods/metods.shtml) de programacion (más adelante doy una pequeña introducción a la filosofía de objetos).

Tercero: Un browser compatible con Java deberá ejecutar cualquier programa hecho en Java, esto ahorra a los usuarios tener que estar insertando "plug-ins" y demás [programas](http://www.monografias.com/Computacion/Programacion/) que a veces nos quitan tiempo y espacio en disco.

Cuarto: Java es un lenguaje y por lo tanto puede hacer todas las cosas que puede hacer un [lenguaje de programación](http://www.monografias.com/Computacion/Programacion/): Cálculos matemáticos,[procesadores](http://www.monografias.com/trabajos5/sisope/sisope.shtml) de palabras, [bases de datos](http://www.monografias.com/trabajos11/basda/basda.shtml), aplicaciones gráficas, animaciones, [sonido](http://www.monografias.com/trabajos5/elso/elso.shtml), hojas de [cálculo](http://www.monografias.com/trabajos7/caes/caes.shtml), etc.

Quinto: Si lo que me interesa son las páginas de [Web](http://www.monografias.com/trabajos5/laweb/laweb.shtml), ya no tienen que ser estáticas, se le pueden poner toda clase de elementos [multimedia](http://www.monografias.com/trabajos10/mmedia/mmedia.shtml) y permiten un alto nivel de interactividad, sin tener que gastar en paquetes carísimos de [multimedia](http://www.monografias.com/trabajos10/mmedia/mmedia.shtml).  
  
  
Todo esto suena muy bonito pero también se tienen algunas limitantes:

La [velocidad](http://www.monografias.com/trabajos13/cinemat/cinemat2.shtml#TEORICO). Los programas hechos en Java no tienden a ser muy rápidos, supuestamente se está trabajando en mejorar esto. Como los programas de Java son interpretados nunca alcanzan la [velocidad](http://www.monografias.com/trabajos13/cinemat/cinemat2.shtml#TEORICO) de un verdadero ejecutable.

Java es un lenguaje de [programación](http://www.monografias.com/Computacion/Programacion/). Esta es otra gran limitante, por más que digan que es orientado a objetos y que es muy fácil de aprender sigue siendo un lenguaje y por lo tanto aprenderlo no es cosa fácil. Especialmente para los no programadores.

Java es nuevo. En pocas palabras todavía no se conocen bien todas sus capacidades.  
Pero en general Java posee muchas ventajas y se pueden hacer cosas muy interesantes con esto. Hay que prestar especial atención a lo que está sucediendo en el mundo de la [computación](http://www.monografias.com/Computacion/index.shtml), a pesar de que Java es relativamente nuevo, posee mucha [fuerza](http://www.monografias.com/trabajos12/eleynewt/eleynewt.shtml) y es tema de [moda](http://www.monografias.com/trabajos16/marketing-hoy/marketing-hoy3.shtml) en cualquier medio computacional. Muchas personas apuestan a futuro y piensan en Java. La pregunta es : ¿Estarán en lo correcto? La verdad es que no se, pero este [manual](http://www.monografias.com/trabajos13/mapro/mapro.shtml) no es para filosofar sobre el futuro del lenguaje sino para aprender a programarlo.

MÁS VENTAJAS

- El JDK es una herramienta libre de licencias (sin costo), creada por Sun.- Está respaldado por un gran número de proveedores.

- Existe soporte dado por Sun.

- Debido a que existen diferentes productos de Java, hay más de un proveedor de servicios.

- Sun saca al mercado cada 6 meses una nueva versión del JDK.

- Es independiente de la plataforma de desarrollo.

- Existen dentro de su librería clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes altamente configurables y con una arquitectura independiente de la plataforma.

- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.

- El conocimiento sobre tecnología Java está en alto crecimiento en el mercado.

- Se puede acceder a bases de datos fácilmente con JDBC, independientemente de la plataforma utilizada.o El manejo de las bases de datos es uniforme, es decir transparente y simple.

- Existen las herramientas Crystal Reports o herramientas libres como iText que los genera en formato pdf.o La API que utilizan estas herramientas en Java, es la más recomendable para generar reportes en Web.

- Simple. Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. Orientado a Objetos. La filosofía de [programación orientada a objetos](http://www.monografias.com/trabajos/objetos/objetos.shtml) es diferente a la[programación](http://www.monografias.com/Computacion/Programacion/) convencional.

- Familiar. Como la mayoría de los programadores están acostumbrados a programar en C o en C++, el sintaxis de [Java](http://www.monografias.com/trabajos16/java/java.shtml) es muy similar al de estos.

- Robusto. El [sistema](http://www.monografias.com/trabajos11/teosis/teosis.shtml) de [Java](http://www.monografias.com/trabajos16/java/java.shtml) maneja [la memoria](http://www.monografias.com/trabajos16/memorias/memorias.shtml) de [la computadora](http://www.monografias.com/trabajos15/computadoras/computadoras.shtml) por ti. No te tienes que preocupar por apuntadores, [memoria](http://www.monografias.com/trabajos13/memor/memor.shtml) que no se esté utilizando, etc. [Java](http://www.monografias.com/trabajos16/java/java.shtml) realiza todo esto sin necesidad de que uno se lo indique.

- Seguro. El [sistema](http://www.monografias.com/trabajos11/teosis/teosis.shtml) de Java tiene ciertas [políticas](http://www.monografias.com/trabajos10/poli/poli.shtml) que evitan se puedan codificar [virus](http://www.monografias.com/trabajos5/virus/virus.shtml) con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los [recursos](http://www.monografias.com/trabajos4/refrec/refrec.shtml) críticos de una [computadora](http://www.monografias.com/trabajos15/computadoras/computadoras.shtml).

- Portable. Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier [computadora](http://www.monografias.com/trabajos15/computadoras/computadoras.shtml) que tenga implementado el interprete de Java.

- Independiente a la [arquitectura](http://www.monografias.com/trabajos6/arma/arma.shtml). Al compilar un programa en Java, el código resultante un tipo de código binario conocido como byte code. Este códido es interpretado por diferentes [computadoras](http://www.monografias.com/trabajos15/computadoras/computadoras.shtml) de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una [arquitectura](http://www.monografias.com/trabajos6/arma/arma.shtml) computacional definida.

- Multithreaded. Un lenguaje que soporta multiples threads es un lenguaje que puede ejecutar diferentes líneas de código al mismo [tiempo](http://www.monografias.com/trabajos6/meti/meti.shtml).

- Interpretado. Java corre en máquina virtual, por lo tanto es interpretado.

- Dinámico. Java no requiere que compiles todas las clases de un programa para que este funcione. Si realizas una modificación a una clase Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

Nota:Java puede funcionar como una aplicación sola o como un "applet", que es un pequeño programa hecho en Java. Los applets de Java se pueden "pegar" a una página de [Web](http://www.monografias.com/trabajos5/laweb/laweb.shtml) ([HTML](http://www.monografias.com/trabajos7/html/html.shtml)), y con esto puedes tener un programa que cualquier [persona](http://www.monografias.com/trabajos7/perde/perde.shtml) que tenga un browser compatible podrá usar.

DESVENTAJAS

- Hay diferentes tipos de soporte técnico para la misma herramienta, por lo que el análisis de la mejor opción se dificulta

- Para manejo a bajo nivel deben usarse métodos nativos, lo que limita la portabilidad.

- El diseño de interfaces gráficas con awt y swing no es simple.

- Existen herramientas como el JBuilder que permiten generar interfaces gráficas de manera sencilla, pero tienen un costo adicional.

- Puede ser que no haya JDBC para bases de datos poco comerciales.

- Algunas herramientas tienen un costo adicional

# ELEMENTOS DEL LENGUAJE JAVA

La sintáxis de un lenguaje define los elementos de dicho lenguaje y cómo se combinan para formar un programa. Los elementos típicos de cualquier lenguaje son los siguientes:

* Identificadores: los nombres que se dan a las variables
* Tipos de datos
* Palabras reservadas: las palabras que utiliza el propio lenguaje
* Sentencias
* Bloques de código
* Comentarios
* Expresiones
* Operadores

A lo largo de las páginas que siguen examinaremos en detalle cada uno de estos elementos.

 Identificadores

Un identificador es un nombre que identifica a una variable, a un método o función miembro, a una clase. Todos los lenguajes tienen ciertas reglas para componer los identificadores:

* Todos los identificadores han de comenzar con una letra, el carácter subrayado ( \_ ) o el carácter dollar ( $ ).
* Puede incluir, pero no comenzar por un número
* No puede incluir el carácter espacio en blanco
* Distingue entre letras mayúsculas y minúsculas
* No se pueden utilizar las plabras reservadas como identificadores

Además de estas restricciones, hay ciertas convenciones que hacen que el programa sea más legible, pero que no afectan a la ejecución del programa. La primera y fundamental es la de encontrar un nombre que sea significativo, de modo que el programa sea lo más legible posible. El tiempo que se pretende ahorrar eligiendo nombres cortos y poco significativos se pierde con creces cuando se revisa el programa después de cierto tiempo.

|  |  |  |
| --- | --- | --- |
| **Tipo de identificador** | **Convención** | **Ejemplo** |
| nombre de una clase | Comienza por letra mayúscula | String, Rectangulo, CinematicaApplet |
| nombre de función | comienza con letra minúscula | calcularArea, getValue, setColor |
| nombre de variable | comienza por letra minúscula | area, color, appletSize |
| nombre de constante | En letras mayúsculas | PI, MAX\_ANCHO |

 Comentarios

Un comentario es un texto adicional que se añade al código para explicar su funcionalidad, bien a otras personas que lean el programa, o al propio autor como recordatorio. Los comentarios son una parte importante de la documentación de un programa. Los comentarios son ignorados por el compilador, por lo que no incrementan el tamaño del archivo ejecutable; se pueden por tanto, añadir libremente al código para que pueda entenderse mejor.

La programación orientada a objetos facilita mucho la lectura del código, por lo que lo que no se precisa hacer tanto uso de los comentarios como en los lenguajes estructurados. En Java existen tres tipos de comentarios

* Comentarios en una sola línea
* Comentarios de varias líneas
* Comentarios de documentación

Como podemos observar un comentario en varias líneas es un bloque de texto situado entre el símbolo de comienzo del bloque /\*, y otro de terminación del mismo \*/. Teniendo encuentra este hecho, los programadores diseñan comentarios como el siguiente:

/\*--------------------------------|

| (C) Angel Franco García |

| fecha: Marzo 1999 |

| programa: PrimeroApp.java |

|---------------------------------\*/

Los comentarios de documentación es un bloque de texto situado entre el símbolo de comienzo del bloque /\*\*, y otro de terminación del mismo \*/. El programa javadoc utiliza estos comentarios para generar la documentación del código.

/\*\* Este es el primer programa de una

serie dedicada a explicar los fundamentos del lenguaje Java \*/

Habitualmente, usaremos comentarios en una sola línea **/**/, ya que no tiene el inconveniente de aprendernos los símbolos de comienzo y terminación del bloque, u olvidarnos de poner este último, dando lugar a un error en el momento de la compilación. En la ventana de edición del Entorno Integrado de Desarrollo (IDE) los comentarios se distinguen del resto del código por el color del texto.

public class PrimeroApp{

public static void main(String[] args) {

//imprime un mensaje

System.out.println("El primer programa");

}

}

Un procedimiento elemental de depuración de un programa consiste en anular ciertas sentencias de un programa mediante los delimitadores de comentarios. Por ejemplo, se puede modificar el programa y anular la sentencia que imprime el mensaje, poniendo delante de ella el delimitador de comentarios en una sola línea.

//System.out.println("El primer programa");

Al correr el programa, observaremos que no imprime nada en la pantalla.

La sentencia System.out.println() imprime un mensaje en la consola, una ventana DOS que se abre en el escritorio de Windows 95. La función println tiene un sólo argumento una cadena de caracteres u objeto de la [clase String.](http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/string.htm)

 Sentencias

Una sentencia es una orden que se le da al programa para realizar una tarea específica, esta puede ser: mostrar un mensaje en la pantalla, declarar una variable (para reservar espacio en memoria), inicializarla, llamar a una función, etc. Las sentencias acaban con ;. este carácter separa una sentencia de la siguiente. Normalmente, las sentencias se ponen unas debajo de otras, aunque sentencias cortas pueden colocarse en una misma línea. He aquí algunos ejemplos de sentencias

int i=1;

import java.awt.\*;

System.out.println("El primer programa");

rect.mover(10, 20);

En el lenguaje Java, los caracteres espacio en blanco se pueden emplear libremente. Como podremos ver en los sucesivos ejemplos, es muy importante para la legibilidad de un programa la colocación de unas líneas debajo de otras empleando tabuladores. El editor del IDE nos ayudará plenamente en esta tarea sin apenas percibirlo.

 Bloques de código

Un bloque de código es un grupo de sentencias que se comportan como una unidad. Un bloque de código está limitado por las llaves de apertura **{** y cierre **}.** Como ejemplos de bloques de código tenemos la definición de una clase, la definición de una función miembro, una sentencia iterativa **for**, los bloques **try ... catch**, para el tratamiento de las excepciones, etc.

 Expresiones

Una expresión es todo aquello que se puede poner a la derecha del operador asignación =. Por ejemplo:

x=123;

y=(x+100)/4;

area=circulo.calcularArea(2.5);

Rectangulo r=new Rectangulo(10, 10, 200, 300);

La primera expresión asigna un valor a la variable x.

La segunda, realiza una operación

La tercera, es una llamada a una función miembro calcularArea desde un objeto circulo de una clase determinada

La cuarta, reserva espacio en memoria para un objeto de la clase Rectangulo mediante la llamada a una función especial denominada constructor.

 Variables

Una variable es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha variable. Hay varios tipos de variables que requieren distintas cantidades de memoria para guardar datos.

Todas las variables han de declararse antes de usarlas, la declaración consiste en una sentencia en la que figura el tipo de dato y el nombre que asignamos a la variable. Una vez declarada se le podrá asignar valores.

Java tiene tres tipos de variables:

* de instancia
* de clase
* locales

Las variables de instancia o miembros dato como veremos más adelante, se usan para guardar los atributos de un objeto particular.

Las variables de clase o miembros dato estáticos son similares a las variables de instancia, con la excepción de que los valores que guardan son los mismos para todos los objetos de una determinada clase. En el siguiente ejemplo, PI es una variable de clase y radio es una variable de instancia. PI guarda el mismo valor para todos los objetos de la clase Circulo, pero el radio de cada círculo puede ser diferente

class Circulo{

static final double PI=3.1416;

double radio;

//...

}

Las variables locales se utilizan dentro de las funciones miembro o métodos. En el siguiente ejemplo area es una variable local a la función calcularArea en la que se guarda el valor del área de un objeto de la clase Circulo. Una variable local existe desde el momento de su definición hasta el final del bloque en el que se encuentra.

class Circulo{

//...

double calcularArea(){

double area=PI\*radio\*radio;

return area;

}

}

En el lenguaje Java, las variables locales se declaran en el momento en el que son necesarias. Es una buena costumbre inicializar las variables en el momento en el que son declaradas. Veamos algunos ejemplos de declaración de algunas variables

**int** x=0;

**String** nombre="Angel";

**double** a=3.5, b=0.0, c=-2.4;

**boolean** bNuevo=true;

**int[]** datos;

Delante del nombre de cada variable se ha de especificar el tipo de variable que hemos destacado en letra negrita. Las variables pueden ser

* Un tipo de dato primitivo
* El nombre de una clase
* Un array

El lenguaje Java utiliza el conjunto de caracteres Unicode, que incluye no solamente el conjunto ASCII sino también carateres específicos de la mayoría de los alfabetos. Así, podemos declarar una variable que contenga la letra ñ

int año=1999;

Se ha de poner nombres significativos a las variables, generalmente formados por varias palabras combinadas, la primera empieza por minúscula, pero las que le siguen llevan la letra inicial en mayúsculas. Se debe evitar en todos los casos nombres de variables cortos como xx, i, etc.

double radioCirculo=3.2;

Las variables son uno de los elementos básicos de un programa, y se deben

* Declarar
* Inicializar
* Usar

Tipos de datos primitivos

|  |  |
| --- | --- |
| Tipo | Descripcion |
| **boolean** | Tiene dos valores **true** o **false**. |
| **char** | Caracteres Unicode de 16 bits  Los caracteres alfa-numéricos son los mismos que los ASCII con el bit alto puesto a 0. El intervalo de valores va desde 0 hasta 65535 (valores de 16-bits sin signo). |
| **byte** | Tamaño 8 bits.  El intervalo de valores va desde -27 hasta 27 -1 (-128 a 127) |
| **short** | Tamaño 16 bits.  El intervalo de valores va desde -215 hasta 215-1 (-32768 a 32767) |
| **int** | Tamaño 32 bits.  El intervalo de valores va desde -231 hasta 231-1 (-2147483648 a 2147483647) |
| **long** | Tamaño 64 bits. El intervalo de valores va desde -263 hasta  263-1 (-9223372036854775808 a 9223372036854775807) |
| **float** | Tamaño 32 bits. Números en coma flotante de simple precisión. Estándar IEEE 754-1985  (de 1.40239846e–45f a 3.40282347e+38f) |
| **double** | Tamaño 64 bits. Números en coma flotante de doble precisión. Estándar IEEE 754-1985. (de 4.94065645841246544e–324d  a 1.7976931348623157e+308d.) |

Los tipos básicos que utilizaremos en la mayor parte de los programas serán **boolean**, **int** y **double**.

### Caracteres

En Java los caracteres no están restringidos a los ASCII sino son Unicode. Un carácter está siempre rodeado de comillas simples como 'A', '9', 'ñ', etc. El tipo de dato **char** sirve para guardar estos caracteres.

Un tipo especial de carácter es la secuencia de escape, similares a las del lenguaje C/C++, que se utilizan para representar caracteres de control o caracteres que no se imprimen. Una secuencia de escape está formada por la barra invertida (\) y un carácter. En la siguiente tabla se dan las secuencias de escape más utilizadas.

|  |  |
| --- | --- |
| **Carácter** | **Secuencia de escape** |
| retorno de carro | \r |
| tabulador horizontal | \t |
| nueva línea | \n |
| barra invertida | \\ |

### Variables booleanas

En el lenguaje C/C++ el valor 0 se toma como falso y el 1 como verdadero. En el lenguaje Java existe el tipo de dato **boolean**. Una variable booleana solamente puede guardar uno de los dos posibles valores: true (verdadero) y false (falso).

boolean encontrado=false;

{...}

encontrado=true;

 Variables enteras

Una variable entera consiste en cualquier combinación de cifras precedidos por el signo más (opcional), para los positivos, o el signo menos, para los negativos. Son ejemplos de números enteros:

12, -36, 0, 4687, -3598

Como ejemplos de declaración de variable enteras tenemos:

int numero=1205;

int x,y;

long m=30L;

int es la palabra reservada para declarar una variable entera. En el primer caso, el compilador reserva una porción de 32 bits de memoria en el que guarda el número 1205. Se accede a dicha porción de memoria mediante el nombre de la variable, numero. En el segundo caso, las porciones de memoria cuyos nombres son x e y, guardan cualquier valor entero si la variable es local o cero si la variable es de instancia o de clase. El uso de una variable local antes de ser convenientemente inicializada puede conducir a consecuencias desastrosas. Por tanto, declarar e inicializar una variable es una práctica aconsejable.

En la tercera línea 30 es un número de tipo **int** por defecto, le ponemos el sufijo **L** en mayúsculas o minúsculas para indicar que es de tipo **long.**

Existen como vemos en la tabla varios tipos de números enteros (**byte**, **short**, **int**, **long**), y también existe una clase denominada BigInteger cuyos objetos pueden guardar un número entero arbitrariamente grande.

 Variables en coma flotante

Las variables del tipo float o **double** (coma flotante) se usan para guardar números en memoria que tienen parte entera y parte decimal.

double PI=3.14159;

double g=9.7805, c=2.9979e8;

El primero es una aproximación del número real , el segundo es la aceleración de la gravedad a nivel del mar, el tercero es la velocidad de la luz en m/s, que es la forma de escribir  2.9979 108. El carácter punto '.', separa la parte entera de la parte decimal, en vez del carácter coma ',' que usamos habitualmente en nuestro idioma.

Otras ejemplos son los siguientes

float a=12.5f;

float b=7f;

double c=7.0;

double d=7d;

En la primera línea 12.5 lleva el sufijo **f**, ya que por defecto 12.5 es **double**. En la segunda línea 7 es un entero y por tanto 7f es un número de tipo **float**. Y así el resto de los ejemplos.

Conceptualmente, hay infinitos números de valores entre dos números reales. Ya que los valores de las variables se guardan en un número prefijado de bits, algunos valores no se pueden representar de forma precisa en memoria. Por tanto, los valores de las variables en coma flotante en un ordenador solamente se aproximan a los verdaderos números reales en matemáticas. La aproximación es tanto mejor, cuanto mayor sea el tamaño de la memoria que reservamos para guardarlo. De este hecho, surgen las variables del tipo **float** y double. Para números de precisión arbitararia se emplea la clase BigDecimal.

Valores constantes

Cuando se declara una variable de tipo final, se ha de inicializar y cualquier intento de modificarla en el curso de la ejecución del programa da lugar a un error en tiempo de compilación.

Normalmente, las constantes de un programa se suelen poner en letras mayúsculas, para distinguirlas de las que no son constantes. He aquí ejemplos de declaración de constantes.

final double PI=3.141592653589793;

final int MAX\_DATOS=150;

## Las cadenas de caracteres o strings

Además de los ocho tipos de datos primitivos, las variables en Java pueden ser declaradas para guardar una instancia de una clase, como veremos en el siguiente capítulo ([Clases y objetos](http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm#Los objetos)).

Las [cadenas de caracteres o strings](http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/string.htm) son distintas en Java y en el lenguaje C/C++, en este último, las cadenas son arrays de caracteres terminados en el carácter  **'\0**'. Sin embargo, en Java son objetos de la clase String.

String mensaje="El primer programa";

Empleando strings, el primer programa quedaría de la forma equivalente

|  |
| --- |
| public class PrimeroApp{  public static void main(String[] args) {  //imprime un mensaje  String mensaje="El primer programa";  System.out.println(mensaje);  }  } |

En una cadena se pueden insertar caracteres especiales como el carácter tabulador '\t' o el de nueva línea '\n'

String texto="Un string con \t un carácter tabulador y \n un salto de línea";

Palabras reservadas

En el siguiente cuadro se listan las palabras reservadas, aquellas que emplea el lenguaje Java, y que el programador no puede utilizar como [identificadores](http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/primero.htm#Identificadores). Algunas de estas palabras le resultarán familiares al programador del lenguaje C/C++. Las palabras reservadas señaladas con un arterisco (\*) no se utilizan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abstract | boolean | break | byte | byvalue\* |
| case | cast\* | catch | char | class |
| const\* | continue | default | do | double |
| else | extends | false | final | finally |
| float | for | future\* | generic\* | goto\* |
| if | implements | import | inner\* | instanceof |
| int | interface | long | native | new |
| null | operator\* | outer\* | package | private |
| protected | public | rest\* | return | short |
| satatic | super | switch | synchronized | this |
| throw | transient | true | try | var\* |
| void | volatile | while |  |  |

Las palabras reservadas se pueden clasificar en las siguientes categorías:

* Tipos de datos: **boolean, float, double, int, char**
* Sentencias condicionales: **if, else, switch**
* Sentencias iterativas: **for, do, while, continue**
* Tratamiento de las excepciones: **try, catch, finally, throw**
* Estructura de datos: **class, interface, implements, extends**
* Modificadores y control de acceso: **public, private, protected, transient**
* Otras: **super**, **null, this**.

OPERADORES LÓGICOS PRINCIPALES EN JAVA

En Java disponemos de los operadores lógicos habituales en lenguajes de programación como son “es igual”, “es distinto”, menor, menor o igual, mayor, mayor o igual, and (y), or (o) y not (no). La sintaxis se basa en símbolos como veremos a continuación y cabe destacar que hay que prestar atención a no confundir == con = porque implican distintas cosas.

|  |  |
| --- | --- |
| OPERADOR | DESCRIPCIÓN |
| == | Es igual |
| != | Es distinto |
| <, <=, >, >= | Menor, menor o igual, mayor, mayor o igual |
| && | Operador and (y) |
| || | Operador or (o) |
| ! | Operador not (no) |

Operadores lógicos principales en Java

 El operador || se obtiene en la mayoría de los teclados pulsando ALT GR + 1, es decir, la tecla ALT GR y el número 1 simultáneamente.

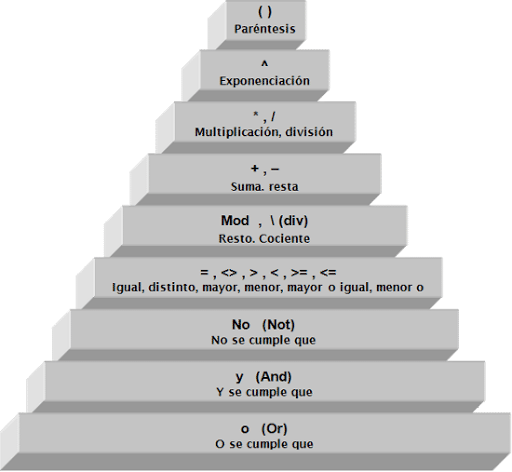
Los operadores && y || se llaman operadores en cortocircuito porque si no se cumple la condición de un término no se evalúa el resto de la operación. Por ejemplo: (a == b && c != d && h >= k) tiene tres evaluaciones: la primera comprueba si la variable a es igual a b. Si no se cumple esta condición, el resultado de la expresión es falso y no se evalúan las otras dos condiciones posteriores.

En un caso como ( a < b || c != d || h <= k) se evalúa si a es menor que b. Si se cumple esta condición el resultado de la expresión es verdadero y no se evalúan las otras dos condiciones posteriores.

El operador ! recomendamos no usarlo hasta que se tenga una cierta destreza en programación. Una expresión como (!esVisible) devuelve false si (esVisible == true), o true si (esVisible == false). En general existen expresiones equivalentes que permiten evitar el uso de este operador cuando se desea.

 ORDEN DE PRIORIDAD, PRELACIÓN O PRECEDENCIA

Los operadores lógicos y matemáticos tienen un orden de prioridad o precedencia. Este es un esquema general que indica el orden en que deben evaluarse en la mayoría de los lenguajes de programación:



Una expresión como A+B == 8 && A-B == 1 siendo A = 3 y B = 5 supondrá que se evalúa primero A+B que vale 8, luego se evalúa A-B que vale -2. Luego se evalúa si se cumple que la primera operación es cierta y luego si la segunda también es cierta, resultando que no, por lo que la expresión es falsa.

#### ESTRUCTURA BÁSICA DE JAVA

Java tiene una estructura interesante que se debe de entender antes de poder comenzar a programar. Para entender esto, veamos el programa el post pasado de este curso de programación.

public class Main  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Hello World!");   
    }  
}

Como pueden ver, aqui tenemos siete lineas de código, con differentes usos. Primero, tenemos la primera linea: public class Main. Public y Class son calificadores que pueden ignorar y dar por hechos, por ahora, pero si se fijan, Main (o como se llame en su programa) tiene el mismo nombre que el archivo. ¿Esto que significa? Con esto podemos ver que todo el programa tiene que ir dentro esta sección. Las secciones se pueden ver definidas por dos llaves "{ }", de la cual una marca el inicio y otra el final. En este archivo, CASI todo esta dentro de las llaves (Eclipse aveces genera código arriba de esto, algo como "package nombre;" no lo borren. Es necesario).

Dentro de eso, se ve otra sección que se llama public static void main(String[] args). Este no se llama main a causa del nombre del archivo, sino es obligatorio. Todo lo que esta en las llaves consiguientes es el código que se va a correr. Otra manera de poner esto es que, todo lo que pongas dentro del main (con m minúscula) son las acciones que el programa va a tomar acabo.

Finalmente, vemos System.out.println("Hello World!");. Esta es una acción del programa. Lo que dice es que "Imprime en la consola el mensaje que esta en parentesis", que en este caso es "Hello World!". Primeramente, necesito que vean unas cosas. Primeramente, que el "Hello World!" esta en comillas. Esto significa que es texto que tu estas dando directamente para que lo imprima. No estas haciendo operaciones ni cambios al mensaje, solo es un valor constante.

También noten que Java es sensible a las mayúsculas, por lo cual system.out.println no es valido pero System.out.println si. Finalmente, vean que esta "oración" (o línea de código) termina en un punto y coma (;). Esto que significa? Veran, en Java podrias omitir compleatamente los espacios y nuevas lineas. Lo único que diferencia una linea de codigo de otra es el punto y coma y las llaves. Ahora, no estoy diciendo que eliminen todos los enters, pero tomen en cuenta que para eso funcionan los ;. Tambien vean como las lineas que son seguidas de llave no llevan punto y coma. Esto sucede con todas las lineas seguidas por llaves.

## 

## CONCLUSIÓN

En el siguiente trabajo pudimos estudiar lo que fue el Lenguaje Java, fue muy elemental investigar en sí que es java , y por qué se caracteriza por ser un lenguaje muy utilizado. Java como sabemos es un lenguaje que fue creado a similitud del lenguaje C pero con una diferencia que este lenguaje nos ayuda a tener una mayor seguridad en el programa que se está ejecutando.

Mediante este lenguaje logramos crear estilos de programa sin la necesitar de aplicar [base de datos](http://www.monografias.com/trabajos34/base-de-datos/base-de-datos.shtml), este programa es útil para [eventos](http://www.monografias.com/trabajos13/gaita/gaita.shtml) de ordenar una [biblioteca](http://www.monografias.com/trabajos10/ponency/ponency.shtml) [personal](http://www.monografias.com/trabajos11/fuper/fuper.shtml), ingresando datos o la idea y lógica del programa es aplicable a cualquier otro tipo de necesidad. Es aquel que sirve para el desarrollo de aplicaciones por lo general [gráficas](http://www.monografias.com/trabajos11/estadi/estadi.shtml#METODOS) , ver un [video](http://www.monografias.com/trabajos10/vire/vire.shtml), o algún [juego](http://www.monografias.com/trabajos15/metodos-creativos/metodos-creativos.shtml) la mayor parte de los celulares lo utilizan.

BIBLIOGRAFIA

Fuentes:

* <http://www.qualitrain.com.mx/objeIndirecto/javavsvbasic.htm>
* [www.monografias.com/trabajos/lengprog/lengprog.shtml](http://www.monografias.com/trabajos/lengprog/lengprog.shtml)
* <http://www.monografias.com/trabajos99/proyecto-crear-archivos-java/proyecto-crear-archivos-java.shtml#ixzz3neCLnC9R>
* <http://www.academica.mx/blogs/lenguaje-programacion-en-java-1>