Introducción al lenguaje Java

En este tema se intenta dar una visión general del lenguaje de programación java. Empezamos con un poco de historia y seguimos con sus características más importantes.

1. EL LENGUAJE DE PROGRAMACIÓN JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. Su creador, **James Gosling**, lo bautizó como **Oak.**
Sun deseaba un lenguaje para programar pequeños dispositivos electrónicos (electrodomésticos y otros aparatos electrónicos de consumo. La dificultad de estos dispositivos es que cambian continuamente y para que un programa funcione en el siguiente dispositivo aparecido, hay que reescribir el código. Por eso Sun quería **crear un lenguaje independiente del dispositivo.**
Se intentaba con este lenguaje paliar uno de los problemas fundamentales del C++: al compilar se produce un fichero ejecutable cuyo código sólo vale para la plataforma en la que se realizó la compilación.
La aparición en 1994 del **navegador gráfico Mosaic** hizo que Internet se popularizara.
Sun se dio cuenta de que los logros conseguidos con su proyecto eran perfectamente aplicables a Internet. Básicamente Internet es una gran red mundial que conecta ordenadores con distintos sistemas operativos y distintas arquitecturas. Esta idea hizo que se abandonara el proyecto de desarrollar un lenguaje común para dispositivos electrónicos de consumo y dirigieran sus investigaciones hacia el desarrollo de un lenguaje que permitiera crear aplicaciones que se ejecutaran en cualquier ordenador de Internet.
En**1995** Oak pasa a llamarse **Java**. Java debe su nombre a un tipo de café. En EEUU se conoce como Java al café, tomarse una taza de Java es tomarse una taza de café. De ahí que el logotipo oficial de Java es una taza humeante de café.
Ese año se da a conocer al público y adquiere notoriedad rápidamente, casi desde su lanzamiento. Se comienza a hablar de Java y de sus applets. Un applet es un programa Java que se ejecuta en el contexto de una página web en cualquier ordenador independientemente de su Sistema Operativo y de la arquitectura de su procesador.
El entorno de ejecución era relativamente seguro y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustadas en las páginas web.
Durante estos años se ha mejorado y se le ha revisado. La versión 1.2 modificó tanto Java que se la llamó **Java 2** y también a sus descendientes (Java 1.3 y Java 1.4). Actualmente el número 2 se ha quitado del nombre y la última versión se conoce como **Java v7**.

2. CARACTERÍSTICAS DEL LENGUAJE JAVA

SENCILLO

Elimina la complejidad de los lenguajes como C y da paso al contexto de los lenguajes modernos orientados a objetos. Aunque la sintaxis de Java es muy similar a C++, elimina algunas de las características más conflictivas de este lenguaje, entre ellas:
- No hay punteros.
- No hay sobrecarga operadores.
- No permite la herencia múltiple.
- No hay necesidad de liberar memoria manualmente. La gestión de memoria dinámica se hace automáticamente (recolector de basura).

ORIENTADO A OBJETOS

Es un lenguaje orientado a objetos puro. En Java todo, a excepción de los tipos fundamentales de variables (int, char, double...) es un objeto.

MULTIPLATAFORMA

Para eliminar la dependencia de la máquina, en Java un programa no se traduce directamente a código ejecutable.

Un programa Java (.java) se compila y se obtiene un código llamado bytecode (.class).

El bytecode lo interpreta la **Máquina Virtual de Java** (JVM).

También se conoce como **JRE** (**Java Runtime Environment**, entorno de ejecución de Java).

El JRE o la máquina virtual de Java se distribuye gratuitamente para prácticamente todos los sistemas operativos, lo que significa que un archivo .class se puede ejecutar en cualquier ordenador o máquina que incorpore el JRE.

Los bytecodes se interpretan por diferentes computadoras de igual manera, por lo que únicamente hay que implementar una máquina virtual para cada plataforma.

El programador compila una única vez el programa Java, y el fichero de bytecode que obtiene se ejecuta igual por la máquina virtual de Java de cualquier plataforma (Windows, Linux, MacOS, etc). De esa forma Java logra ser un lenguaje que no depende de una arquitectura de ordenador específica.

La ejecución mediante la máquina virtual hace que la ejecución de los programas Java sea más lenta que la de los programas escritos en C++.

Para aumentar la velocidad de ejecución se pueden emplear:
- **Compiladores de código nativo**: el programa java se compila y se obtiene directamente código máquina ejecutable. Perdemos la portabilidad.

- **Compiladores JIT (Just-In-Time)**: A medida que se van interpretando los bytecodes se guarda el código máquina generado en un buffer. De esta forma no es necesario volver a traducir de nuevo el código ya traducido con lo que ganamos en velocidad.

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 empleado en el desarrollo de aplicaciones Java.

SEGURO

Al contrario de lo que sucede con C/C++, no se puede acceder a la memoria mediante punteros.
Además Java incorpora medidas que evitan que se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no se puede hacer con los recursos críticos de una computadora.

MULTITAREA

Soporta múltiples *threads*, hilos o tareas. Esto quiere decir que puede ejecutar diferentes líneas de código al mismo tiempo tanto si la máquina es multiprocesador como si no lo es.

DINAMICO

En Java no es necesario cargar completamente el programa en memoria sino que las clases compiladas pueden ser cargadas bajo demanda en tiempo de ejecución (dynamic binding).
Esto proceso permite la carga de código bajo demanda, lo que es especialmente importante en los applets.

3. TIPOS DE APLICACIONES QUE SE PUEDEN DESARROLLAR CON JAVA

APPLETS

Son programas Java pensados para ser colocados dentro de una página web. Pueden ser interpretados por cualquier navegador con capacidades Java. Estos programas se insertan en las páginas usando una etiqueta especial (como también se insertan vídeos, animaciones flash u otros objetos).
Hoy día mediante applets se pueden integrar en las páginas web aplicaciones multimedia avanzadas (incluso con imágenes 3D o sonido y vídeo de alta calidad)

SERVLETS

Son aplicaciones que se ejecutan en un servidor de aplicaciones.

APLICACIONES DE CONSOLA

Son programas independientes al igual que los creados con los lenguajes tradicionales.

APLICACIONES GRÁFICAS

Aquellas que utilizan las clases de Java con capacidades gráficas.

MIDLET

Aplicación creada para su ejecución en dispositivos móviles. Por ejemplo, los juegos Java creados para teléfonos móviles.

4. LA PLATAFORMA JAVA

Una plataforma es el ambiente de software o hardware en el que corre un programa.

La plataforma Java consta de dos componentes:

- La máquina virtual de Java

- La API de Java (Application Programming Interface, interfaz de programación de aplicaciones)

La API de Java es una colección de bibliotecas con clases estándar.

Estas clases se pueden incluir en los programas Java, sin temor a fallos de portabilidad. Además, están bien documentadas (mediante páginas Web), y organizadas en paquetes y en un gran árbol de herencia.

A este **conjunto de paquetes** se le conoce como la **API** de Java.

Los paquetes básicos de la API de Java son:

PAQUETES DE UTILIDADES

java.lang: Fundamental para el lenguaje. Incluye clases como String o StringBuffer.
java.io: Para la entrada y salida a través de flujos de datos, y ficheros del sistema.
java.util: Contiene colecciones de datos y clases, el modelo de eventos, facilidades horarias, generación aleatoria de números, y otras clases de utilidad.
java.math: Clases para realizar aritmética con la precisión que se desee.
java.text: Clases e interfaces para manejo de texto, fechas, números y mensajes de una manera independiente a los lenguajes naturales.
java.security: Clases e interfaces para seguridad en Java: Encriptación RSA...

PAQUETES PARA EL DESARROLLO GRÁFICO

java.applet: Para crear applets y clases que las applets utilizan para comunicarse con su contexto.
java.awt: Para crear interfaces con el usuario, y para dibujar imágenes y gráficos.
javax.swing: Conjunto de componentes gráficos que funcionan igual en todas las plataformas que Java soporta.
javax.accesibility: Da soporte a clases de accesibilidad para personas discapacitadas.
java.beans: Para el desarrollo de JavaBeans.

PAQUETES PARA EL DESARROLLO EN RED

java.net: Clases para aplicaciones de red.
java.sql: Paquete que contiene el JDBC, para conexión de programas Java con Bases de datos.
java.rmi: Paquete RMI, para localizar objetos remotos, comunicarse con ellos e incluso enviar objetos como parámetros de un objeto a otro.
org.omg.CORBA: Facilita la posibilidad de utilizar OMG CORBA, para la conexión entre objetos distribuidos, aunque esté codificados en distintos lenguajes.
org.omb.CosNaming : Da servicio al IDL de Java, similar al RMI pero en CORBA
Actualmente hay tres ediciones de Java. Cada una de ellas se corresponde con una plataforma que incluye una serie de funciones, paquetes y elementos del lenguaje (la API).
JAVA SE. Java Standard Edition.
Antes se la conocía como **J2SE**. Permite escribir código Java relacionado con la creación de aplicaciones y applets en lenguaje Java común.
JAVA EE. Java Enterprise Edition.
Todavía conocida como **J2EE**. Pensada para la creación de aplicaciones Java empresariales y del lado del servidor.
Java ME. Java Micro Edition.

También conocida como **J2ME**. Pensada para la creación de aplicaciones Java para dispositivos móviles, PDAs y otros dispositivos electrónicos.

5. El JDK

Para poder escribir un programa Java es necesario tener instalado el Kit de Desarrollo de Java o JDK (**Java Development Kit**), también llamado Java SDK (Software Development Kit).

El JDK contiene el software necesario para que los programadores compilen, depuren y ejecuten programas y applets escritos en Java.

Tanto el software como la documentación son gratuitos según el acuerdo de la licencia de Sun Microsystems.

Se puede descargar en:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

En esta página también se puede descargar el **JRE**.

El entorno JDK no es el más adecuado para el desarrollo de aplicaciones Java, debido a funcionar única y exclusivamente **mediante comandos de consola**:

javac Es el comando compilador de Java.

Su sintaxis es:

javac ejemplo.java

La entrada de este comando ha de ser necesariamente un fichero que contenga código escrito en lenguaje Java y con extensión .Java. El comando nos creará un fichero .class por cada clase que contenga el fichero Java.

Los ficheros .class contienen código bytecode, el código que es interpretado por la máquina virtual.

java Es el intérprete de Java.

Permite ejecutar aplicaciones que previamente hayan sido compiladas y transformadas en ficheros .class.

Su sintaxis es:

java ejemplo

No es necesario aquí suministrar la extensión del fichero, ya que siempre ha de ser un fichero .class.

appletviewer Se trata de un comando que verifica el comportamiento de un applet. La entrada del comando ha de ser una página web que contenga una referencia al applet que deseamos probar.

Su sintaxis es:

appletviewer mipagina.html

El comando ignora todo el contenido de la página web que no sean applets y se limita a ejecutarlos.

javadoc Permite generar documentación en formato html sobre el contenido de ficheros con extensión .Java.

Su sintaxis es:

javadoc ejemplo.java

En la documentación generada por este comando se puede ver que métodos y constructores posee una determinada clase, junto con comentarios sobre su uso, la versión, el autor de la clase....

Hay que prestar atención al directorio en el que se ha instalado el JDK. La razón es que debemos **modificar tres variables del sistema**:

PATH

Variable que contiene rutas por defecto a los programas que indiquemos. La razón es que por ejemplo el comando java debe de estar disponible estemos en la carpeta que estemos. Dicho comando (junto con el resto de comandos del JDK) está en la carpeta bin dentro de la carpeta en la que hemos instalado el JDK

JAVA_HOME

Variable utilizada por la mayoría de aplicaciones basadas en Java que contiene la ruta a la carpeta en la que se instaló el JDK.

CLASSPATH

Es una variable similar al PATH que sirve para indicar rutas a las carpetas en las que se almacenarán aplicaciones Java.

La forma de configurar estas variables en Windows es:

Abrir el menú contextual de **Mi PC** (o **Equipo** en Windows Vista, Windows 2008 o Windows 7) y elegir **Propiedades**. Después elegir **Propiedades Avanzadas** y finalmente pulsar en el botón **Variables de entorno**

Seleccionar la variable PATH y pulsar en modificar. **Sin borrar nada de lo que contiene**, debemos añadir al final del texto el símbolo **;** y después la ruta al directorio bin dentro de la carpeta del JDK.

Tras aceptar el cuadro anterior, podremos pulsar en **Nueva** para añadir la variable JAVA_HOME indicando como valor la ruta al JDK.

Podemos comprobar si todo se ha realizado correctamente comprobando la versión de Java. Para comprobar la versión de Java basta con ir al símbolo del sistema Windows y escribir
java –version

6. VERSIONES DEL JDK

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

A partir de la versión 1.2 se habla de Java 2. A partir de la versión 1.6 se deja de utilizar J2SE para llamarse Java SE 6, Java SE 7.

Cada versión tiene varias revisiones, por ejemplo la versión 1.6.7 del JDK hace referencia a la revisión 7 de la versión 6.

7. DOCUMENTACIÓN. API DE JAVA

En http://java.sun.com/docs/ se encuentra la documentación oficial de Java.
La más interesante es la documentación del **API** de Java.

Se puede descargar la documentación en un archivo zip. De esta forma no dependeremos de la conexión a Internet para consultar la documentación oficial. Esto se puede realizar de la página de descarga
http://www.oracle.com/technetwork/java/javase/downloads/index.html

8. ENTORNOS DE DESARROLLO PARA JAVA

El kit de desarrollo básico proporcionado por Sun es lo mínimo que se necesita para desarrollar un programa en Java. Es útil si se necesita compilar aplicaciones Java de manera esporádica

Sin embargo, la escritura y compilación de programas hecha de esta forma es un poco incomoda. Por ello numerosas empresas fabrican sus propios entornos de edición, algunos incluyen el compilador y otras utilizan el propio JDK de Sun.

Algunas ventajas que ofrecen son:

- Facilidades para escribir código.
- Facilidades de depuración.
- Facilidad de configuración del sistema.
- Facilidades para organizar los archivos de código.
- Facilidad para exportar e importar proyectos

Los entornos de desarrollo para Java más utilizados en la actualidad son:

NETBEANS

Uno de los IDE Java más populares. Es un entorno gratuito de código abierto para la generación de código en diversos lenguajes.
Contiene prácticamente todo lo que se suele pedir a un entorno de desarrollo, editor avanzado de código, depurador, diversos lenguajes, extensiones de todo tipo (CORBA, Servlets,...).
Incluye además un servidor de aplicaciones Tomcat para probar aplicaciones de servidor.
Consume bastantes recursos. Tiene una arquitectura extensible con módulos específicos para desarrollo web, aplicaciones móviles, diseño UML, etc.
Puede obtenerse gratuitamente de http://netbeans.org/

ECLIPSE

Junto a netBeans, el entorno de desarrollo Java gratuito más utilizado Es un entorno completo de código abierto.
También permite el desarrollo en C++ y otros lenguajes de programación.
Puede descargarse en http://www.eclipse.org/

BORLAND JBUILDER

Entorno de desarrollo completo creado por la empresa Borland para la creación de todo tipo de aplicaciones Java, incluidas aplicaciones para móviles. Existen versiones limitadas que pueden bajarse de www.borland.com

MICROSOFT VISUAL J#

Uno de los más populares, aunque las aplicaciones obtenidas pueden presentar problemas de compatibilidad con el SDK oficial de Java, por el uso de librerías especificas de Microsoft.
Permite construir aplicaciones Java dentro de la plataforma .NET. La versión express puede obtenerse gratuitamente en http://www.microsoft.com/express