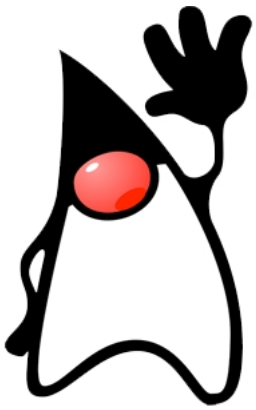
There are no translations available.



Java es un lenguaje de Programación Orientado a Objetos que ha tenido gran auge debido, principalmente, a su estrecha relación con internet pues está especialmente adaptado para facilitar la realización de aplicaciones para la Word Wide Web (www).

Sirve, como los demás lenguajes de programación, para definir como debe comportarse el ordenador, que tareas debe hacer ante las acciones del usuario o de otros sistemas, y puede ser utilizado en cualquier campo desde aplicaciones de gestión (de bancos, de personal, etc), hasta aplicaciones control de equipos (coches, electrodomésticos, etc).

Java es un lenguaje desarrollado por la compañía informática Sun Microsystems, una de las mayores del sector.

Su origen está en el desarrollo, a finales de los años 80, de un lenguaje específico para electrodomésticos que requería un interfaz de usuario sencillo e intuitivo. James Gosling que formaba parte del equipo de desarrollo unió su experiencia en un lenguaje que él había estado diseñando, llamado OAK, basado en la sintaxis de C++, con los requisitos del nuevo proyecto. Entre éstos estaba la necesidad de que cuando se realice un programa sobre un microprocesador determinado (por ejemplo sobre un Pentium IV de nuestros días ) no sea necesario modificarlo si lo que se desea es ejecutarlo en otro microprocesador, algo que ocurría habitualmente con todos los lenguajes de programación (C, Pascal, Fortran, ADA, )

Este problema era especialmente importante en el campo de la electrónica de consumo pues los fabricantes de electrodomésticos hacían sus equipos (lavadoras, microondas, etc) con un microprocesador dado, y sobre el realizaban la programación pertinente para los programas de lavado, de cocción, del funcionamiento en general del equipo, etc. Si surgía un nuevo procesador (mejor, más barato), todos los fabricantes lo incluían en sus cadenas de producción pues las grandes tiradas de aparatos suponían un gran ahorro de costes, pero a cambio debían rehacer los programas de los distintos aparatos. La idea era tener un lenguaje independiente de la plataforma.

Como resultado de estas inquietudes se realizó un proyecto de sistema de control completo de los aparatos electrónico de una casa y otro para realizar televisión interactiva en el lenguaje Java primitivo pero el escaso éxito comercial de ambos dejo relegado al lenguaje a un segundo término. En 1995, gracias a la proliferación de internet y al intento de la compañía Sun de disputar su primacía en el sector informático a Microsoft se realizó una actualización del lenguaje Java para adaptarlo a internet manteniendo su idea de diseño fundamental de independencia de la plataforma. Gracias a su inclusión en navegadores como Netscape (Agosto de 1995)que permitían ejecutar paginas web dinámicas realizadas con java el lenguaje ha tenido el auge que conocemos actualmente

Las principales características de java son:

**Es simple**: he eliminado características de otros lenguajes como la necesidad de liberar memoria, la utilización de punteros, etc.

**Es orientado a objetos**: Durante el desarrollo de la informática se ha pasado por diversas fases en la realización de los programas, desde la programación con 0 y 1 hasta la programación estructurada, lenguajes de 4ª generación, etc.

En la orientación a objetos se trata de basar la realización de la aplicación en elementos similares a los del mundo real a los que definimos una serie de operaciones y un estado propio, después la aplicación funcionará por la interacción y relaciones entre los diferentes objetos.

Por ejemplo, si estamos una aplicación para el control de un instituto, primero definiríamos las clases de objetos que intervienen en la misma: Personas, Aulas, Asignaturas, Grupos de alumnos, etc. A cada uno de ellos les asociaríamos sus características intrínsecas (nombre, dni, teléfono, etc a las personas, localización y nombre a las aulas, etc.) y sus operaciones (por ejemplo  Poner una Nota sería una de las operaciones que puede realizar un profesor)

Además los lenguajes orientados a objetos tienen otras características como la Herencia (que permite que una vez definida una clase de objetos (por ejemplo Persona) se pueda reutilizar sin tener que volver a hacerla para crear otras clases más específicas (como el Alumno o el Profesor que aunque siguen siendo personas y poseen las características definidas para las mismas (nombre, dni, etc) poseen algunas más (Curso y asignaturas en las que está matriculado el Alumno).,  Otras características Orientadas a Objetos son la encapsulación y el polimorfismo cuya explicación excede la pretensión de este artículo.

**Facilita la realización de aplicaciones distribuidas**: que son aquellas que tienen partes de la misma ejecutando en varios ordenadores que se comunican entre si para que el funcionamiento sea el correcto

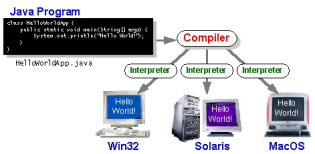
**Es robusto**: Detecta problemas de los programas antes de tener que ejecutar éstos y no permite que se utilicen instrucciones poco fiables.

**Es Seguro**: No permite accesos a zonas de la memoria del ordenador de forma indiscriminada, ni la sustitución de una parte del programa por otra ajena (caballo de troya) , ni, por ejemplo, que cuando se abre una aplicación, que está en un ordenador remoto, en el ordenador local de usuario, no se puedan modificar datos del ordenador local

**Es Independiente de la Plataforma**: Característica ya comentada anteriormente y que se consigue debido a que la compilación, proceso por el que se consigue que el programa que escribe el programador (programa fuente) sea convertido en un programa capaz de entender y ejecutar la máquina (programa ejecutable), se realiza en 2 pasos:

En un primer paso se crea un fichero llamado ByteCode (fichero con extensión .class) a partir

del programa fuente. Este ByteCode puede ser llevado a cualquier ordenador que tenga una Máquina Virtual Java para ser transformado en un ejecutable y ser ejecutado. La Máquina Virtual (MV), que es específica para cada tipo de ordenador, es una aplicación que permite traducir el ByteCode genérico a las características concretas del ordenador en el que se está ejecutando.




**Es interpretado**: No se crea directamente el código ejecutable sino que se va ejecutando y traduciendo poco a poco el programa a ejecutar. Este proceso lo lleva a cabo la JVM que en la figura haría el papel del interprete. Esto hace que los programas en java sean más lentos que los realizados en otros lenguajes.

**Permite ejecuciones en paralelo de varias tareas. (multithreaded)**

**Es dinámico**: Si durante la ejecución de una aplicación java falta una de las piezas de la misma, tiene mecanismos para buscarla en la red y traerla automáticamente para su uso.

**Instalación del SDK**:

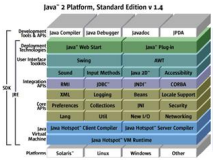
Para realizar un programa en java es necesario disponer de un entorno que se encargue de traducir lo que nosotros escribimos en algo comprensible por el ordenador.

Existen muy diversos tipos de entorno, algunos son gratis y otros no.

Vamos a centrarnos aquí en uno de los más ampliamente utilizados que es el J2SE que se puede obter

http://java.sun.com/j2se/1.4.2/download.html . Después de elegir la versión correspondiente a tu sistem

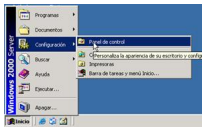
Para realizar la instalación deberás ejecutar el fichero resultado de la descarga y seguir las instrucciones que



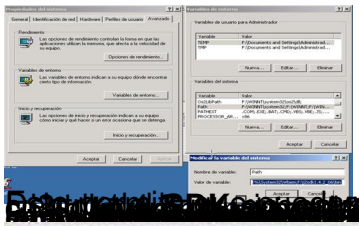

También encontrarás unas instrucciones básicas de instalación en la página de descarga: http://java.sun.com/j2se/1.4.2/install.html
:

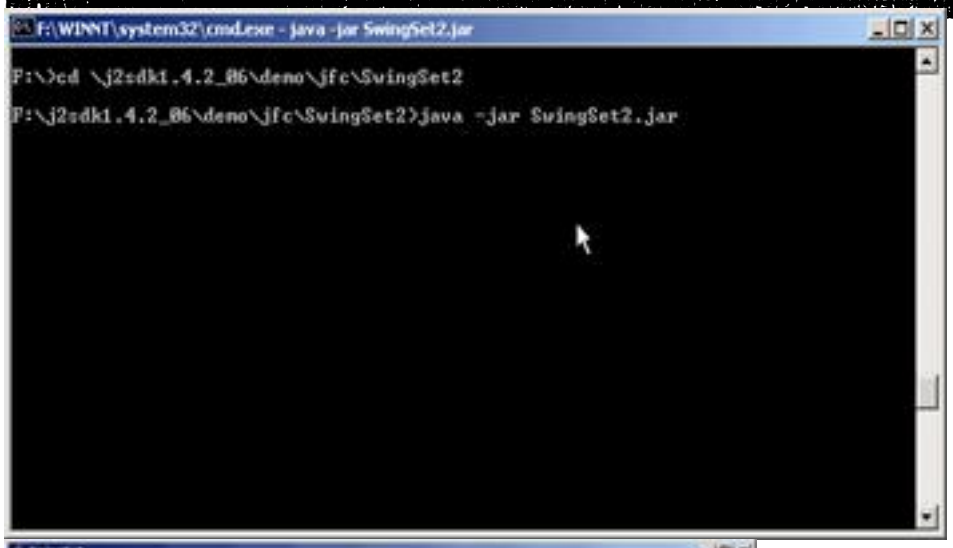
Lo más importante de las mismas consiste en colocar adecuadamente el valor de la variable PATH

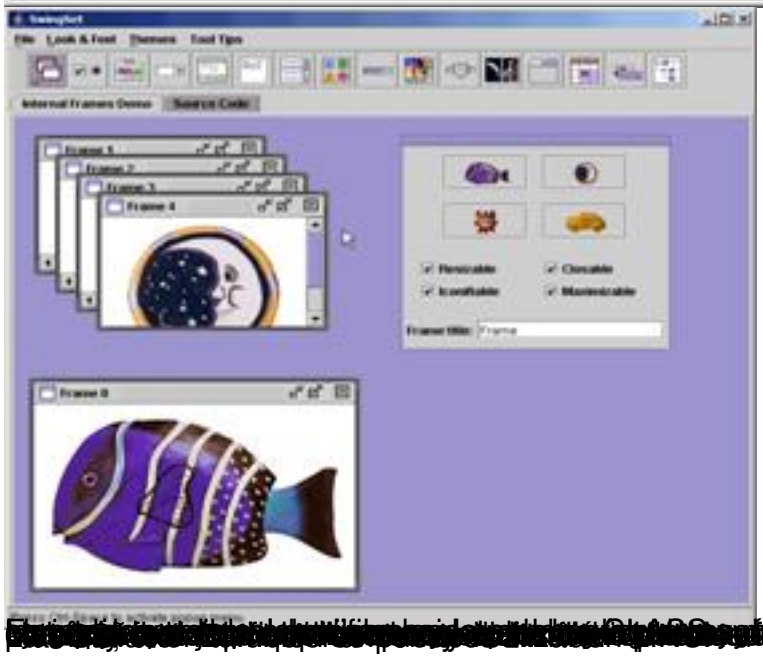
Si tienes un sistema operativo Windows 2000 Pro/XP deberás seleccionar *Sistema* en el menú de inicio de wind




Después seleccionar *Avanzado* y pulsar en *Variables de Entorno*. Buscar la variable *PATH* en las variables del sistema o en las de usuario.
*editarla*
y añadirle el directorio
*bin*
de la versión que hemos bajado después de un punto y coma (;) Es decir la variable PATH podría quedar con un contenido como el siguiente:
*%SystemRoot%system32;%SystemRoot%;%SystemRoot%System32Wbem*
***;C:j2sdk1.4.2_06 in***





Haciendo una distinción, no muy rigurosa pero que puede servir de idea para un usuario novel, podemos dividir en 3 los tipos básicos de aplicaciones que se pueden realizar con java:

- Aplicaciones en modo comando: son las más simples de realizar y las que se deberían practicar en primer lugar para comprender y practicar con las características del lenguaje. No tienen entorno gráfico por lo que se deberán ejecutar en una ventana de comandos ( En windows se abriría una ventana de comandos tipo MSDOS)

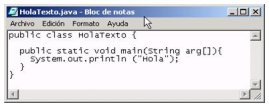
- Aplicaciones con entorno gráfico (Java Swing o AWT) son ejecutadas igual que las anteriores pero utilizan el entorno gráfico por lo es necesario conocer ciertas características de programación en entornos gráficos como el manejo de eventos (sucesos producidos cuando el usuario pulsa un botón de una ventana gráfica o selecciona un elemento de un menú), la creación de objetos gráficos, etc.
- Applets, Aplicaciones embebidas en páginas web. Para ejecutarlas se realiza, además del programa java correspondiente, una página html en la que se hace la llamada a la aplicación y posteriormente se abre dicha página con un navegador web o similar.

Vamos a ver un ejemplo de cada uno de estos tipos.

Primero vamos a crear una aplicación básica suponiendo que no tenemos ningún IDE (entorno de desarrollo integrado) más adelante indicaremos algunos y las direcciones de donde se pueden obtener. Para conseguir algo ejecutable de forma inmediata procederemos como sigue:

1. Abre una ventana de un editor de textos como el notepad ( *Menú Inicio->programas->accesorios->bloc de notas.*

2. Escribe el programa siguiente




Vamos a explicar algunos conceptos básicos:

- En java todos los programas tienen que tener al menos una clase.

Las clases se declaran con la palabra *class* precediendo al nombre de la clase. Entre llaves {} se definirán los elementos que componen la clase que serán atributos que almacenan el estado

de cada objeto creado del tipo de la clase definida (como el tamaño y color de un objeto Ventana) y métodos (operaciones asociadas a los objetos como el cerrar o abrir ventana).

En nuestro ejemplo la clase se denomina *HolaTexto* y tiene un método llamado *main*, no tiene atributos

Por convenio, los nombres de las clases tienen la primera inicial de cada palabra en mayúsculas y todas las palabras unidas sin guiones. ejemplo CocheBomberos y no Coche-Bomberos ni cochebomberos, ni cocheBomberos

- Para poder ejecutar una aplicación es necesario que exista un método llamado *main* que tiene que ser definido literalmente con
*public static void main(String arg[])*
{} dentro de las llaves se colocan las instrucciones que debe ejecutar el ordenador cuando se invoca al programa..

En nuestro caso la única instrucción existente es *System.out.println ("Hola, Buenas tardes.");* que escribe en pantalla
*Hola, Buenas Tardes*

Por convenio los nombres de los métodos tienen la inicial de cada palabra en Mayúsculas excepto la primera y van unidas todas las palabras sin guiones. Por ejemplo *ponerNotasAlumno*

· Todas las instrucciones terminan en punto y coma

Ya podemos pasar a ejecutar el programa:

1. Guarda el fichero, con el nombre *HolaTexto.java*, en uno de tus directorios

4. Colócate en el directorio en el que has guardado el fichero en la ventana de msdos utilizando el comando cd visto anteriormente ( por ejemplo *cd c:MidirJava*)

5. Ejecuta el comando de compilación del fichero

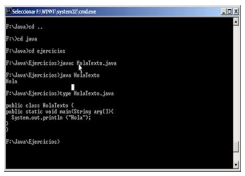
*javac HolaTexto.java*

Si no hay ningún error se creará un fichero HolaTexto.class que es el fichero con el bytecode

1. Ejecuta el comando para ejecutar la aplicación:

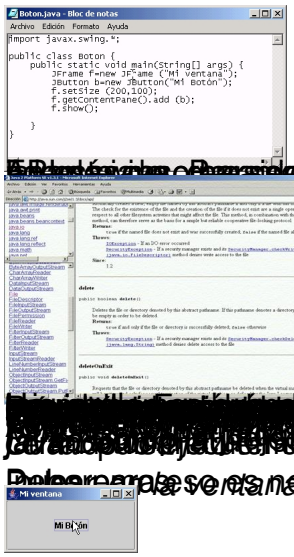
*java HolaTexto*

aparecerá en la pantalla el mensaje Hola, Buenas tardes.



Si deseamos hacer una aplicación gráfica, podemos realizar los mismos pasos que antes con otro programa:

1. Escribe el programa siguiente




*Si queremos un ejemplo más completo en el que además del botón aparece un mensaje en la pantalla que cuenta cuantas veces se pulsa el botón y además permite realizar una acción (mostrar el mensaje Fin del programa ) cuando se cierra la ventana, podemos usar el siguiente:*

```
import javax.swing.*;

import java.awt.event.*;

class EscuchaVentana extends WindowAdapter{

        public void windowClosing(WindowEvent e) {

                System.out.println ("Fin del programa")

                System.exit(0);
```

```
                }

}

public class ContadorBoton extends JFrame{

    private int contador = 0;

    ContadorBoton () {

        super ("Ejemplo swing");

        final JLabel etiqueta= new JLabel("Pulsa el botón");

        JButton boton = new JButton("Botón");

        boton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                contador=contador+1;
```

```
                    etiqueta.setText("Contador: " + contador);

                }

            });

            etiqueta.setLabelFor(boton);

            JPanel panel = new JPanel();

            panel.add(boton);

            panel.add(etiqueta);

            getContentPane().add(panel);

            addWindowListener(new EscuchaVentana() );

            pack();

            setVisible(true);
```

*}*

*public static void main(String[] args) {*

*ContadorBoton cb= new ContadorBoton();*

*}*

*}*

*En el que además del botón hemos incluido la gestión de algunos eventos.*

*En primer lugar se ha creado una claseEscuchaVentana para poder adaptar los eventos que se producen en la ventana a nuestra aplicación (*
*class EscuchaVentana extends WindowAdapter{ } )*
*. Hemos codificado el método*
*windowClosing*
*que se ejecuta cuando se desea cerrar la ventana para que cuando el usuario la cierre se muestre en la pantalla desde la que hemos ejecutado la aplicación el mensaje Fin del programa.*

- *Después hemos indicado que la claseContadorBoton para definir el formato gráfico de la aplicación ( public class ContadorBoton extends JFrame);*

- *Hemos creado un atributo de la claseContadorBoton para contar cuantas veces se pulsa el botón. ( int contador=0;)*

*· Se define un constructor (método que crea un objeto concreto) para definir como se creará la ventana (public ContadorBoton(){ }).*

*y en el se define:*

*super( "Ejemplo Swing " );*

*Defino el título de la Ventana*

*final JLabel etiqueta= new JLabel("Pulsa el botón");*

*Mensaje que se escribirá la 1ª vez que se crea la ventana*

*JButton boton = new JButton("Botón");*

*Creo el botón*

*boton.addActionListener(*

*new ActionListener() {*

*public void ActionPerformed(ActionEvent e) {*

*contador=contador+1;*

*etiqueta.setText("Contador: " + contador);*

*}*

*});*

*Indico que hay que hacer cuando se pulsa el botón (aumentar el contador en 1 )*

*etiqueta.setLabelFor(boton)*

*La etiqueta está asociada al botón*

*JPanel panel = new JPanel();*

*Creo un panel en el que colocaré los elementos (etiqueta y botón) para tener organizada la disposición de los mismos*

*panel.add(boton);*

*Añado el botón al panel*

*panel.add(etiqueta);*

*Añado la etiqueta al panel*

*getContentPane().add(panel);*

*Asocio el panel a mi ventana*

*addWindowListener(new EscuchaVentana() );*

*Se indica que las acciones sobre la ventana (cerrarla, iconizarla,..) las realizará la clase EscuchaVentana definida anteriormente.*

*pack();*

*El tamaño de la ventana se ajustará al tamaño de sus*

*setVisible(true);*

*Hace visible la ventana*

*Después se define el programa principal en el que se crea la ventana (public static void main( String args[] ) { ..} )*

*Si quisiera crear varias ventanas iguales, por ejemplo 2, sólo tendría que poner dentro del método main las instrucciones:*

*ContadorBoton cb= new ContadorBoton();*

*ContadorBoton cb2= new ContadorBoton();*

*Después debo salvar el fichero con el nombre ContadorBoton.java y ejecutar los comandos*

*javac ContadorBoton.java*

*java ContadorBoton*

*Aplicación al ejecutarse*

*Aplicación después de pulsar 3 veces el botón*

*Un applet es una aplicación a la que se accede por medio de un navegador web como el Internet Explorer, Netscape, Mozilla, etc.*

*En lugar de tener un método main que se ejecuta cuando se invoca a la aplicación tiene los métodos*

*void init() que ejecuta el navegador la primera vez que se carga el applet.*

*void start () que se ejecuta cada vez que se carga el applet*

*void stop() que se ejecuta cuando se abandona el documento que contiene el applet*

*void destroy () que se ejecuta cuando el navegador determina que no se va a necesitar más el applet.*

*Para construir un applet además del programa java necesitamos una página html que haga referencia al fichero bytecode que tiene el código del applet (fichero .class).*

*Veamos un ejemplo como quedaría la aplicación anterior que muestra un botón y un mensaje con el número de veces que se ha pulsado el botón transformada en un applet.*

*1.□□□□□□ Crearíamos el ficheroAppletContarBoton.java, basado en el programa anterior y modificado ligeramente para adaptarse a la modelo de un applet, con el contenido siguiente:*

*import javax.swing.*;□□□□□□□□□□□*

```
import java.awt.*;

import java.awt.event.*;

public class AppletContarBoton extends JApplet{

    private int contador = 0;

    JButton boton ;

    JLabel etiqueta;

    public AppletContarBoton () {

        getRootPane().putClientProperty("defeatSystemEventQueueCheck",

                                        Boolean.TRUE);

    }

    public void init () {
```

```
      setContentPane(crearPanel());

    }

    public Container crearPanel() {

        etiqueta= new JLabel("Pulsa el Botón");

        boton = new JButton("Boton");

        boton.setActionCommand("Boton");

        boton.setEnabled(true);

        boton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                contador=contador+1;

                etiqueta.setText("Contador: " + contador);
```

```
                }

        });

        etiqueta.setLabelFor(boton);

        boton.setBounds(20,95, 50,55);

        JPanel panel = new JPanel();

        panel.add(boton);

        panel.add(etiqueta);

        panel.setBackground(Color.green);

        return panel;

    }

    public void actionPerformed(ActionEvent evento){

            contador++;
```

*}*

*}*

*2. Compilariamos dicho fichero para crear el bytecode (AppletContarBoton.java) ejecutando el comando:*

*javac AppletContarBoton.java*

*3. Crearíamos el fichero AppletContarBoton.html con el contenido siguiente:*

*para que cuando se abra esta página html se ejecute nuestro applet*

*4. Visualizaríamos la aplicación. Lo podemos hacer de 2 formas:*

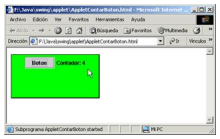
*· Ejecutando desde la ventana de MSDOS el comando*

*appletviewer AppletContarBoton.html*

- *Abriendo la página AppletContarBoton.html con un navegador web (a veces las recargas de los navegadores no funcionan adecuadamente y no recuperan la última versión del applet )*

*appletviewer AppletContarBoton.html*

*Abriendo la página AppletContarBoton.html con el Internet Explorer*



*Entre los más sencillos están:*

- ***Jcreator :*** ***http://www.jcreator.com/***

- ***Eclipse :*** ***http://www.eclipse.org/***

*También se pueden obtener algunos más complejos como:*

- ***Java*** *Studio Enterprise (antiguo Forte) : http://es.sun.com/tecnologia/software/desarrollo_aplicaciones/herramientas/enterprise6/index.html*

- *Netbeans:*

- *Websphere: http://www-106.ibm.com/developerworks/websphere/downloads/WASTD6Support.html*

*Si se utiliza un entorno de los anteriores los pasos básicos de creación de una aplicación son*

1. *Abrir la aplicación*
2. *Elegir nuevo fichero*
3. *Seleccionar Proyecto*
4. *Elegir el tipo de proyecto (aplicación java, applet ,..) y asignarle un nombre, una ubicación, etc.*
5. *Escribir el programa*
6. *Compilarlo con el menú Build/compile*
7. *Ejecutar con Build/execute*