1. INTRODUCCIÓN

Objetivos:
a) Describir las características del lenguaje de programación Java
b) Describir las herramientas ligadas a la construcción y ejecución de programas escritos en Java
c) Construir las primeras aplicaciones en Java

Este capítulo pretende ser una rápida introducción a la programación en Java. En primer lugar muestra lo que es Java, sus características y las herramientas que están ligadas a él y, a continuación, enseña cómo compilar y ejecutar algunos programas sencillos escritos en Java. La tecnología Java es tanto una plataforma como un lenguaje de programación. En los capítulos posteriores se trata de dar una visión más detallada de la sintaxis del lenguaje de programación Java.

1.1. El lenguaje de Programación Java

El lenguaje de programación Java, fue diseñado por la compañía **Sun** Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en sistemas de ordenadores heterogéneos (redes de computadoras formadas por más de un tipo de ordenador, ya sean PC compatibles, Macintosh o estaciones de trabajo que empleen diferentes sistemas operativos como Windows, OS/2 o Unix), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

Su origen se remonta a la creación de un lenguaje de programación para el desarrollo de aplicaciones para electrodomésticos y otros aparatos electrónicos de consumo por parte de una empresa filial de Sun, llamada FirstPerson en 1991. Su creador, **James Gosling**, lo bautizó como *Oak*. Al abandonarse este proyecto, el lenguaje se modificó, al igual que su nombre y se orientó al desarrollo de aplicaciones para la red. En septiembre de 1995 aparece el primer *Kit de Desarrollo de Java* (JDK). A principios de 1997 se presenta la primera revisión de Java (la versión 1.1) y a finales de 1998 surge la versión 1.2 (Java 2) que introdujo modificaciones bastante significativos. En octubre de 2004 se hace pública la versión Java 1.5 (Java 5) incluyendo innovaciones muy importantes en la plataforma.

Características del lenguaje

Según la propia Sun Microsystems, el lenguaje Java muestra las siguientes características generales:

- **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 y C++, que son lenguajes a los que una gran mayoría de programadores están acostumbrados a emplear.
- **Orientado a Objetos**. La filosofía de programación orientada a objetos es diferente a la programación convencional (*imperativa* o *procedural*). Su nivel de abstracción facilita la creación y mantenimiento de programas. Existen muchas referencias que dan una introducción a esta forma de programar.
- **Independiente** a la arquitectura y portable. Al compilar un programa en Java, el código resultante es un tipo de código binario conocido como *Java bytecodes*. Este código es

interpretado por diferentes computadoras de igual manera, por lo que únicamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura de ordenador específica. Como el código compilado de Java es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.

- **Robusto**. Java simplifica la gestión de la memoria dinámica. Por ejemplo, ya no es necesario la liberación explícita, el intérprete de Java lo lleva acabo automáticamente cuando detecta que una variable dinámica ya no es usada por el programa. Por otra parte, impide que un puntero Java apunte a una dirección de memoria no válida, los punteros (referencias) Java son seguros y deterministas: o bien apuntan a un elemento correctamente alojado en memoria o bien tienen el valor nulo. Finalmente el acceso a la memoria es supervisado por el intérprete de tal manera que no es posible acceder a zonas de memoria no autorizadas sin provocar un error. Por ejemplo, no es posible escribir fuera de los límites de un vector.
- **Seguro**. El sistema de Java tiene ciertas políticas que evitan que se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los denominados *applets*, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.
- **Multitarea** (*Multithreaded*). Un lenguaje que soporta múltiples *threads*, hilos o tareas, es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo. El soporte y la programación de hilos en Java está integrado en la propia sintaxis del lenguaje.
- **Dinámico**. 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.

Mecanismo de creación de un programa de Java

En este aspecto la principal originalidad de Java estriba en que es a la vez compilado e interpretado. Con el compilador de Java, el programa fuente con extensión `.java` es traducido a un lenguaje intermedio o pseudo-código (no es código máquina) llamado Java *bytecodes* generándose un programa *compilado* almacenado en un archivo con extensión `.class` . Este archivo puede ser posteriormente *interpretado* y ejecutado por el intérprete de Java (lo que se conoce como la Máquina Virtual Java o *Java Virtual Machine*). Por eso Java es multi-plataforma, ya que existe un intérprete para cada máquina diferente. Por tanto, la compilación se produce una vez y la interpretación cada vez que el programa se ejecuta. Este proceso se esquematiza en la Figura 1.1.

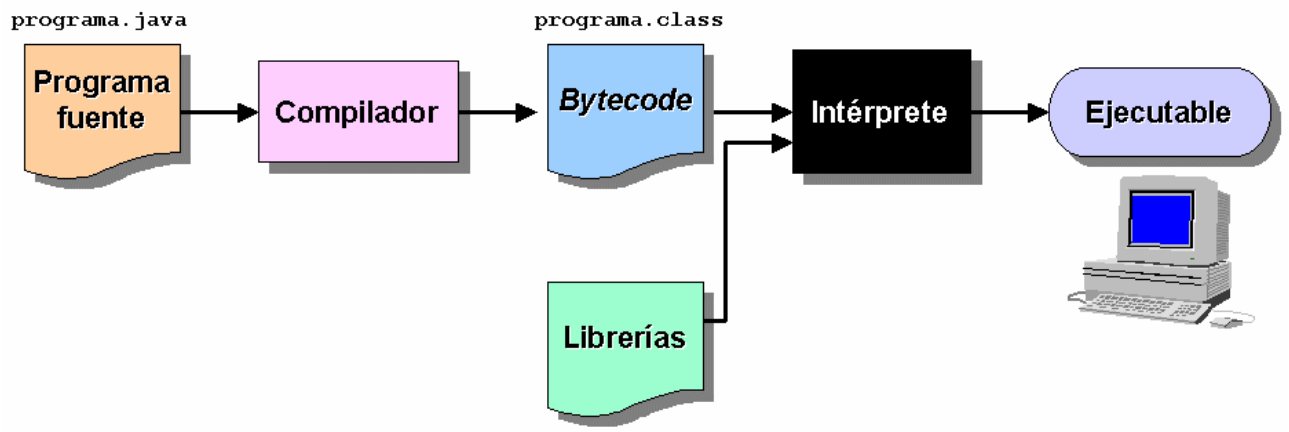

Figura 1.1. Esquema del proceso de creación de un programa con Java

Actualmente las máquinas virtuales modernas realizan una compilación JIT (*Just In Time*) en donde el *bytecode* no es interpretado sino que se compila directamente a código máquina en tiempo de ejecución de acuerdo con la arquitectura (procesador y sistema operativo) en la que se ejecuta la máquina virtual. Esto permite conseguir velocidades de ejecución similares al C. En la práctica las máquinas virtuales suelen utilizar técnicas mixtas de interpretación/compilación JIT normalmente según la frecuencia de paso por un *bytecode* concreto.

Un programa Java puede funcionar como una aplicación independiente (por ejemplo, el entorno de desarrollo *NetBeans*) o como un *applet* (contracción de la expresión *little application*), que es un pequeño programa que no se ejecuta de forma independiente. Los *applets* de Java se pueden introducir o incrustar en una página de Web (empleando el lenguaje HTML), y con esto se puede tener un programa que puede ser ejecutado por cualquier persona que tenga un navegador compatible con Java. Aunque queda fuera del alcance de este manual, es necesario indicar que también pueden construirse un tercer tipo de aplicaciones: los denominados *servlets* (contracción de la expresión *server application*), que se ejecutarían en servidores web conectados a intranets o a internet.

Funcionamiento de un *applet* de Java

El funcionamiento de un *applet* de Java dentro de un documento HTML en el servicio WWW puede esquematizarse en los siguientes pasos:

1. En un servidor Web existe un código de Java almacenado en un archivo con extensión `.class` y un documento HTML que hace referencia a este archivo.
2. Una persona con un navegador compatible con Java realiza una conexión al servidor y la petición del documento HTML anterior.
3. El servidor envía el documento HTML y el archivo con extensión `.class`.
4. Ambos llegan al ordenador cliente y el intérprete ó Máquina Virtual de Java que, en este caso, está en el navegador, transforma el código Java `.class` en un código que entiende la máquina local y el programa correspondiente se ejecuta visualizándose dentro de la página de WWW.
5. Si el usuario realiza otra conexión a otro URL o abandona la sesión del navegador, el programa se deja de ejecutar y en el ordenador no queda rastro de él.

Ventajas en el uso de Java

Pueden destacarse las siguientes ventajas en el empleo de Java como lenguaje de programación:

- **Compatibilidad**. No es necesario modificar (reescribir) el código si se desea ejecutar el programa en otra máquina. Un único código funciona para todos los navegadores compatibles con Java o donde se tenga una Máquina Virtual de Java (ordenadores PC compatibles, Macintosh o estaciones de trabajo que empleen diferentes sistemas operativos como Windows, Mac OS X, Linux o Unix).
- **Funcionalidad**. Si interesa desarrollar un servicio Web con funciones dinámicas más allá de las posibilidades del lenguaje HTML, puede emplearse Java para incluir en las páginas toda clase de elementos multimedia y permitir un alto nivel de interactividad.
- **Ahorro de recursos**. Un navegador compatible con Java deberá ejecutar cualquier programa hecho en Java, esto ahorra a los usuarios tener que estar insertando programas adicionales o *plug-ins* que necesitan emplear memoria adicional y espacio en disco.

- **Metodología OO**. Java es un lenguaje de programación orientado a objetos, y tiene todos los beneficios que ofrece esta metodología de programación: facilita la creación, el mantenimiento y la reutilización de código.
- **Menos y mejor código**. Comparaciones de métricas de programas indican que un programa en escrito en Java es cuatro veces de menor tamaño que uno escrito en C++ y además favorece los buenos hábitos en la programación como, por ejemplo, la gestión de la memoria dinámica.
- **Gratuidad**. El kit de desarrollo Java es gratuito y puede descargarse de diversos servidores WWW y FTP de la red.

Inconvenientes del lenguaje Java

El uso de Java también tiene algunos inconvenientes o limitaciones:

- **Mayor consumo memoria**: un programa Java consume más memoria por dos razones, es necesario cargar la máquina virtual y, en general, Java necesita más memoria para alojar los elementos de un programa que un programa similar hecho en un lenguaje nativo.
- **Mayor tiempo de carga**: la carga de la máquina virtual lleva tiempo y como la carga de las clases son bajo demanda la ejecución al principio de un programa Java es relativamente lenta.
- **Integración no perfecta con el sistema operativo**: como Java y sus librerías están diseñados para ser multiplataforma la integración con el sistema operativo en forma de extensiones al mismo no es sencilla y suele necesitar extensiones nativas que rompen la portabilidad. Por otro lado exigen la presencia y carga de la máquina virtual por lo que no se suele utilizar como lenguaje para el desarrollo de elementos básicos de sistemas.
- **Es un lenguaje de programación**. El hecho de que Java sea un lenguaje de programación es otra gran limitación. Aunque sea orientado a objetos y "más sencillo" de aprender que C o C++, sigue siendo un lenguaje y por lo tanto aprenderlo no es tarea fácil. Especialmente para los programadores noveles.

La plataforma Java

Normalmente, una plataforma es un sistema mixto que incluye el hardware y/o el entorno software en el que se ejecuta un programa. La **plataforma Java** se diferencia de la mayoría de las demás en que está formada únicamente por software que se ejecuta en cualquier otra plataforma independiente de hardware. La plataforma Java tiene dos componentes:

- El intérprete, Máquina Virtual Java ó *Java Virtual Machine* (*Java VM*) que ya se ha comentado anteriormente, y
- La Interfaz de Programación de Aplicaciones Java ó *Java Application Programming Interface* (*Java API*).

El API de Java es una amplia colección de componentes de software que facilitan muchas necesidades de programación como puede ser código necesario para construir una interfaz de usuario (GUI). El API de Java se agrupa en librerías o paquetes (*packages*) de componentes relacionados entre sí: componentes básicos de programación, creación de *applets*, redes, internacionalización, seguridad, componentes de software, conectividad y redes, etcétera. Hay, además, extensiones estándar fuera del núcleo del API de Java que facilitan recursos para servidores, gráficos 3D, animación…

La Figura 1.2 esquematiza la relación entre la aplicación o *applet* de Java, la Máquina Virtual, el API y el *hardware* correspondiente.

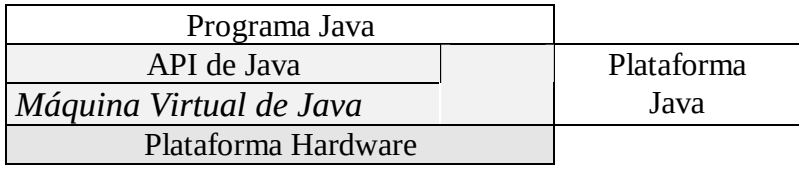

Figura 1.2. Esquema de la relación entre las plataformas en Java

1.2. Instalación del Kit de Desarrollo Java

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

Obtención del archivo de instalación

Si no se posee el archivo de instalación, puede conseguirse de algún CD de software distribuido gratuitamente junto con muchas revistas especializadas o bien, descargarlo de algún servidor FTP anónimo o WWW. En este último caso se pueden obtener más fácilmente las versiones más recientes del kit de desarrollo. En este sentido se recomienda descargar una versión reciente del JDK de la *edición estándar* también llamada *Java SE Development Kit*. La dirección WWW es `http://java.sun.com`

Ejecución del archivo de instalación

Una vez almacenado el archivo de instalación en un directorio local, se ejecuta para comenzar el proceso de instalación del SDK. La instalación se lleva a cabo con la asistencia de una secuencia de ventanas de diálogo que van presentando las distintas etapas de la instalación. En la ventana de confirmación del inicio de la instalación se pide la confirmación correspondiente al usuario, y posteriormente, el programa de instalación del SDK va pidiendo al usuario la confirmación para realizar las distintas etapas de la instalación.

Configuración del JDK

Tras la instalación del entorno, los directorios del software tienen la estructura mostrada en la Figura 1.3. Esto también depende de los componentes seleccionados previamente del JDK.

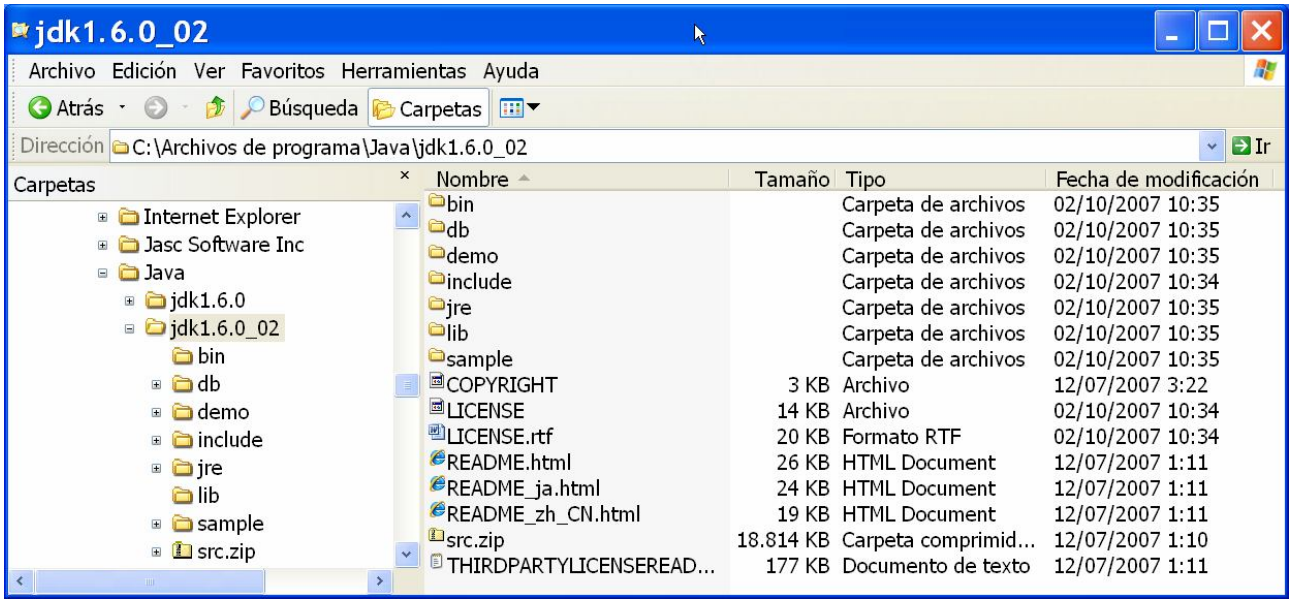

Figura 1.3. Directorio principal del JDK para la versión 1.6.0_02

Actualización de la variable del entorno PATH (opcional)

Es posible utilizar el JDK sin modificar ninguna de la variable del entorno del sistema PATH o sin modificar el archivo AUTOEXEC.BAT. Sin embargo, se recomienda configurar adecuadamente la variable PATH.

> Nota: En caso de tener Windows XP es preferible realizar los siguientes cambios a través del Panel de Control (Inicio/Panel de Control/Sistema/Entorno) en lugar de hacerlo modificando directamente el archivo `autoexec.bat`. Una vez abierto el Panel de control, se selecciona Sistema y se modifican las variables de entorno:

- La variable de entorno PATH. Si se desea utilizar los archivos ejecutables del JDK desde cualquier directorio es necesario especificar el directorio donde se encuentran estos ejecutables en el valor asignado a la variable PATH.

```
PATH = c:\windows;lo que sea… ;C:\Archivos de programa\java\JDK1.5.0\BIN
```

Ejecución de las herramientas del JDK

El sistema está ya configurado y listo para poder emplear el JDK. Puede ejecutarse cualquier aplicación escribiendo su nombre en una ventana DOS con el archivo correspondiente como parámetro. Ninguna de las herramientas Java son programas ejecutables que utilicen una interfaz de usuario gráfico (GUI ó *Graphic User Interface*) tipo Windows. Es decir, no pueden ejecutarse pulsando dos veces sobre su icono desde la ventana de Windows.

1.3. Primeras prácticas

A continuación se dan las indicaciones oportunas para llevar a cabo tres tareas básicas trabajando con Java:

a) la creación de un programa sencillo
b) la inserción de un *applet* ya existente en un documento HTML
c) la creación de un *applet* y su posterior inserción en un documento HTML

Como se ha comentado en el prefacio de este manual, el ordenador y el entorno de programación (compilador, intérprete...) son excelentes maestros. Es altamente recomendable experimentar con cada una de los elementos y características del lenguaje Java que vayan estudiándose y analizar los mensajes de error y de advertencia obtenidos al compilar y ejecutar los programas.

Creación del programa "Hola, me llamo Angel"

Para escribir el primer programa se necesita lo siguiente:

a. la plataforma Java incluida en el Kit de Desarrollo de Java.
b. Un editor de texto. Un programa Java se escribe utilizando el sistema de codificación de caracteres que emplea 16 bits llamado Unicode. Los primeros 127 caracteres de este sistema coinciden con el código ASCII. Las herramientas de desarrollo Java actuales leen los caracteres ASCII y los convierten sobre la marcha a Unicode. Si se está utilizando el sistema operativo Windows puede emplearse el editor de texto Block de notas (Inicio -> Programas -> Accesorios -> Block de notas) para escribir un programa fuente de Java (Figura 1.4).

Siguiendo los siguientes pasos puede crearse un programa Java (ejecutable de forma independiente en cualquier Máquina Virtual Java) que, al ejecutarse, visualice un mensaje por pantalla.

1. Crear un archivo fuente Java. Utilizando un editor de texto, crear un archivo llamado `Hola.java` con el siguiente código escrito en el lenguaje de programacion Java (cuidado: hay que respetar las mayúsculas y las minúsculas en el nombre del archivo y en su contenido tal y como se escribe a continuacion):

```
/**
 *  La clase hola construye un programa que visualiza el mensaje
 *  "hola, me llamo Angel" en el dispositivo de salida estandar
 */
public class Hola {
    public static void main(String[] args) {
         // Visualiza por pantalla "Hola, me llamo Angel"
         System.out.println("Hola, me llamo Angel");
    }
}
```

Atención: Es muy importante respetar las mayúsculas y las minúsculas en el nombre del archivo y en su contenido tal y como aparece en el código anterior, ya que tanto el compilador como el intérprete de Java (a diferencia de lo que ocurre con otros lenguajes de programación) son sensibles a las mayúsculas y minúsculas. En este sentido, `Hola` es distinto de `hola`.

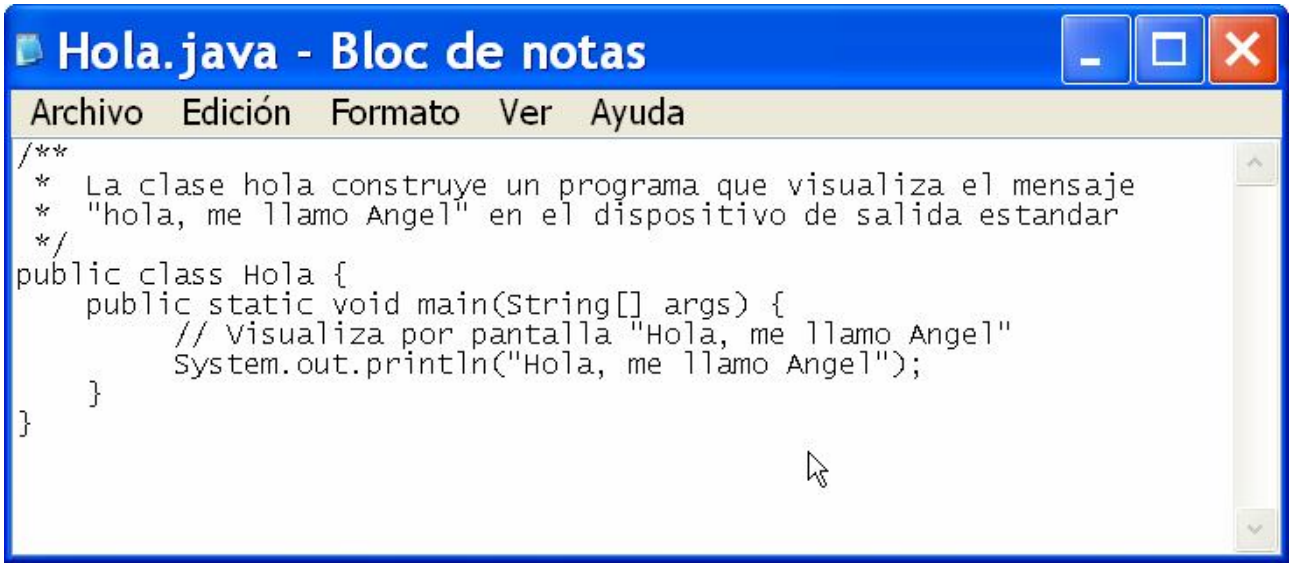

Figura 1.4. Ventana del Block de notas de Windows para la edición del programa fuente

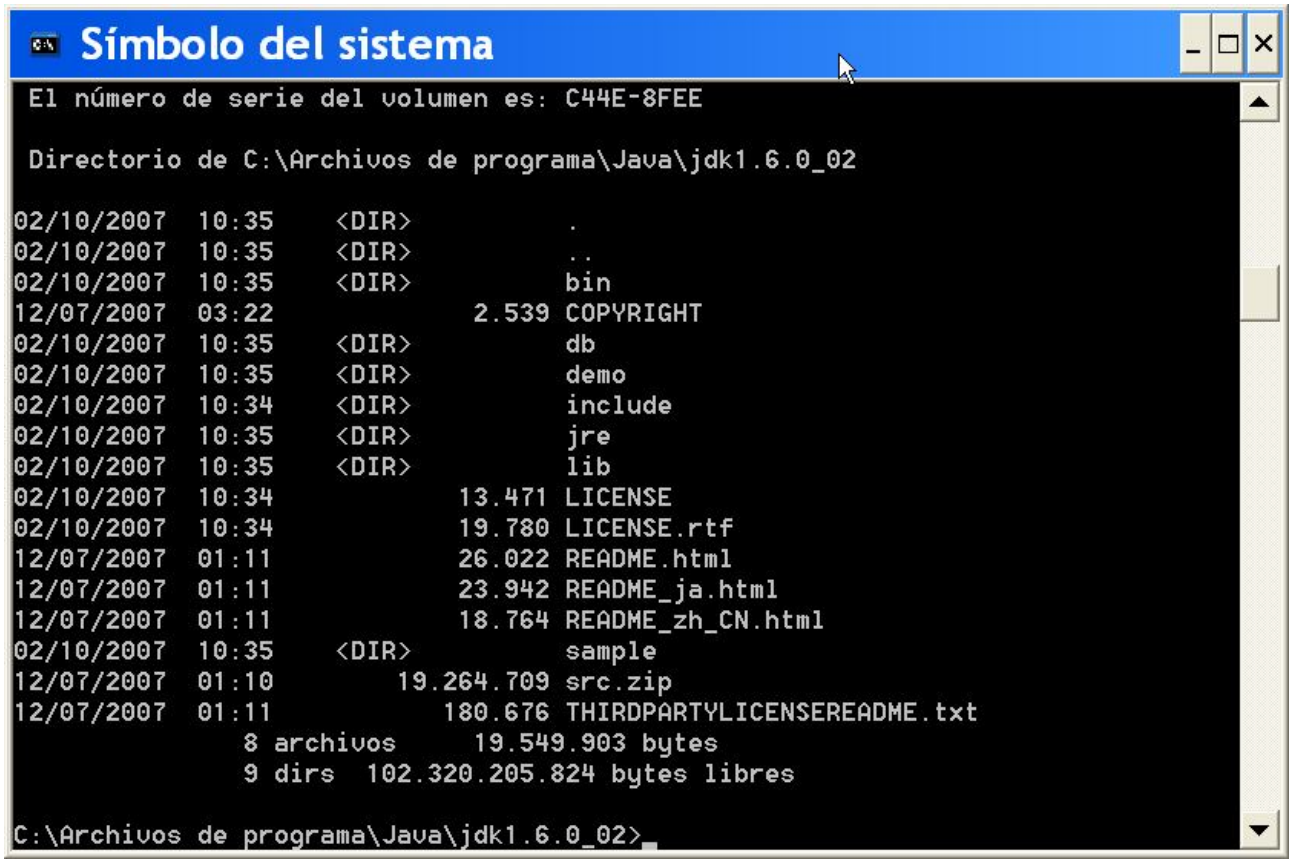

Figura 1.5. Ventana del sistema DOS con el contenido del directorio actual de trabajo

2. Compilar el archivo fuente utilizando el compilador de Java (`javac.exe`):

```
$>javac Hola.java↵
```

Nota importante: cuando el compilador indica un error de sintaxis, el error podría no estar en la línea que indica el mensaje de error de compilación. Primero, se debe comprobar la línea en donde se indica que existe un error. Si esta línea no contiene el error, debe verificarse el código de las líneas anteriores del programa.

Si la compilación tiene éxito, el compilador crea un archivo llamado `Hola.class` en el mismo directorio que el archivo fuente `Hola.java`. El archivo `.class` contiene los *bytecodes* de Java que es un código independiente de la plataforma.

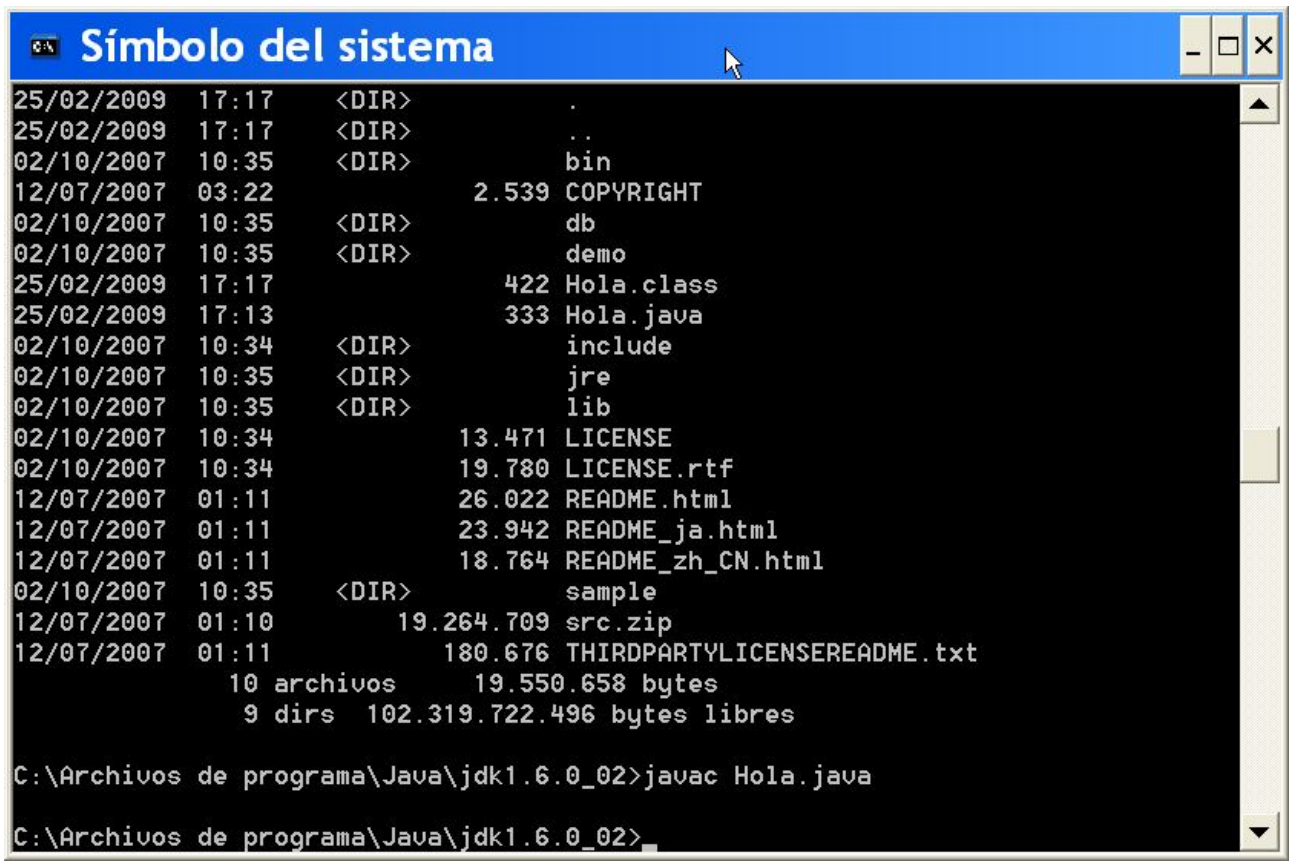

Figura 1.6. Ventana del sistema DOS con la línea de comandos para la ejecución de programas

3. Ejecución del programa utilizando el intérprete de Java (`java.exe`):

```
$>java Hola↵
```

Si todo va bien debería visualizarse en pantalla (Figura 1.7) el mensaje: `Hola, me llamo Angel`.

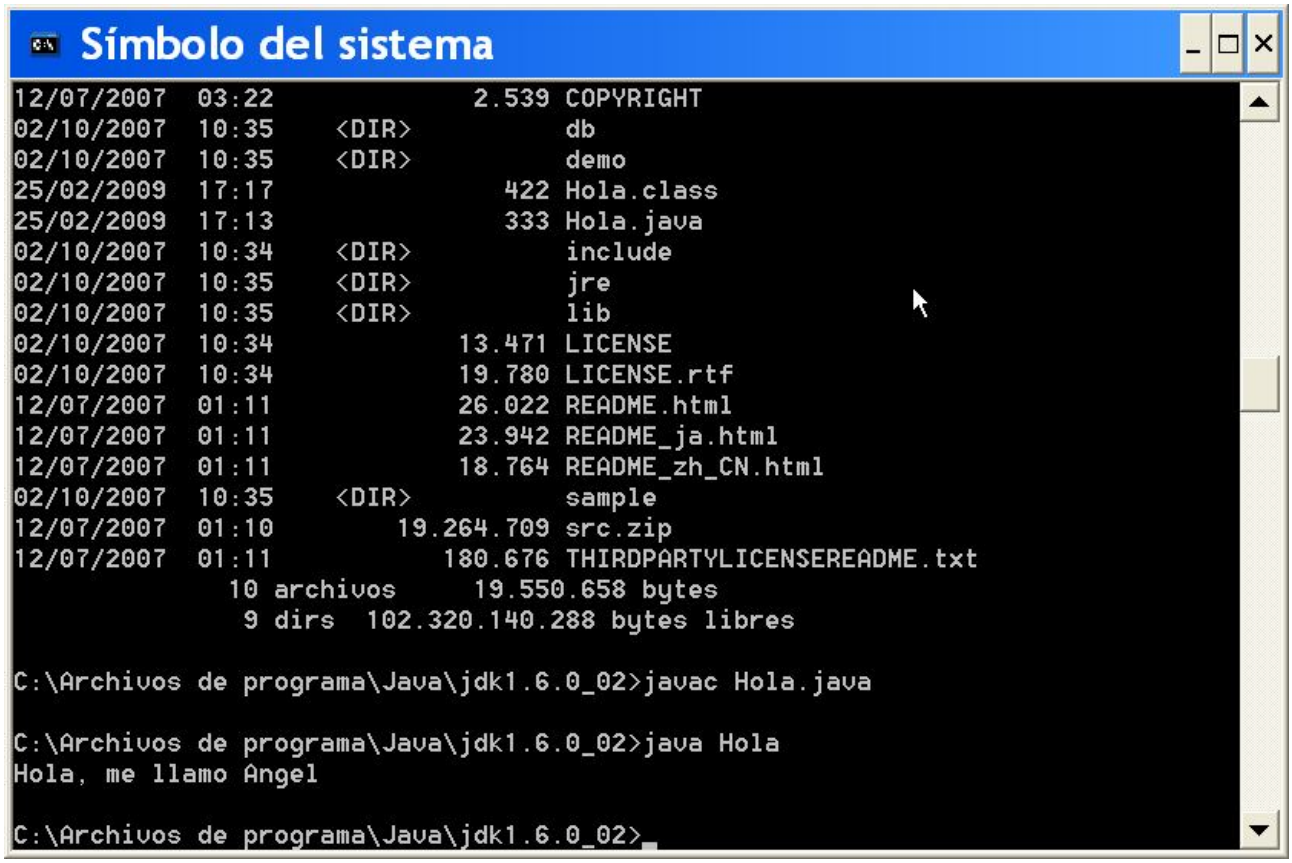

Figura 1.7. Ventana del sistema DOS con la salida por pantalla al ejecutar el programa

Convención para las extensiones de los archivos

El *software* de Java emplea las extensiones para los archivos indicadas en la Tabla 1.1.

Tabla 1.1. Tipo de archivo y extensión correspondiente

Tipo de archivo	Extensión
Fuente de Java	`.java`
bytecode de Java	`.class`

Introducción de un *applet* de Java en un documento HTML

El elemento `applet` permite incluir pequeñas aplicaciones (*applets*) escritas con código Java en documentos HTML. Dentro del elemento `applet` se puede indicar un texto alternativo o un enlace a otro documento para los navegadores que no admitan código Java. Esta alternativa puede reducirse a un mensaje de texto que es ignorado por los navegadores que sí admiten Java. Un ejemplo de elemento `applet` tomado de Rachel Gollub[1] que se reconstruye según se muestra en la Figura 1.8 es el siguiente:

```
<html>
  <head>
```

[1] `http://java.sun.com/applets/jdk/1.4/demo/applets/Clock/example1.html` (Visitado por última vez el 27 de febrero de 2009)

```
    <title>Documento con applet de reloj</title>
  </head>
  <body>
    <!-- Este es el cuerpo del documento HTML -->
    <applet code="Clock2.class" width="175" height="160">
     Este texto se visualiza si el navegador
     no admite Applets
    </applet>
  </body>
</html>
```

La visualización del documento HTML se realiza con un navegador WWW que tenga instalado un intérprete de Java (Figura 1.8).

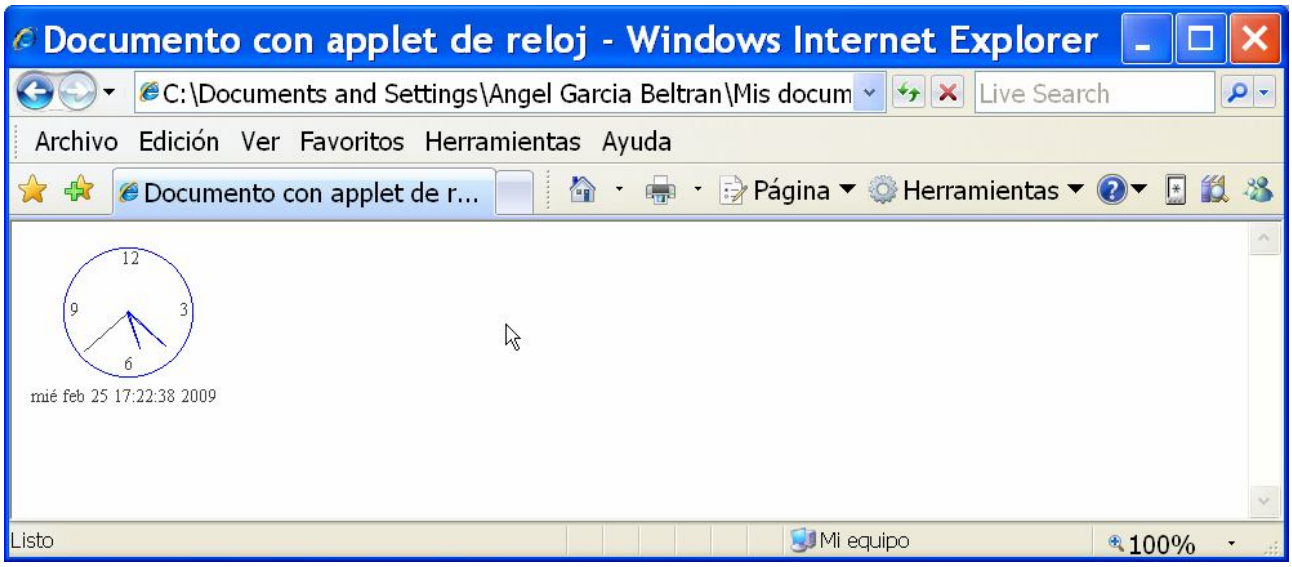

Figura 1.8. Visualización de un documento HTML con un *applet* de Java

El elemento `param` se incluye dentro del elemento `applet` con el fin de transferir datos o parámetros a la aplicación Java. La sintaxis del elemento PARAM, que se compone exclusivamente de una etiqueta de apertura, es:

```
<param name="Parámetro" value="valor">
```

Los atributos de la etiqueta son `name` que sirve para especificar el nombre del parámetro de la aplicación y `value` para establecer su valor. No todos los *applets* de Java admiten parámetros. Un ejemplo de elemento `applet` que emplea parámetros es el siguiente:

```
<applet code="Clock2.class" width=165 height=140>
    <param name=bgcolor value="000000">
    <param name=fgcolor1 value="FFFF00">
    <param name=fgcolor2 value="FFFFFF">
  Este texto se visualiza si el navegador
  no admite Applets
</applet>
```

En este caso particular, el *applet* `Clock2` admite tres parámetros: el color de fondo (`bgcolor`), el color de la corona y de las manecillas (`fgcolor1`) y el color del segundero y de los

dígitos (`fgcolor2`). Los valores de estos parámetros son códigos RGB hexadecimales. En el ejemplo anterior se les asocia los colores negro (`000000`), amarillo (`FFFF00`) y blanco (`FFFFFF`), respectivamente. El resultado es el que se muestra en la Figura 1.9.

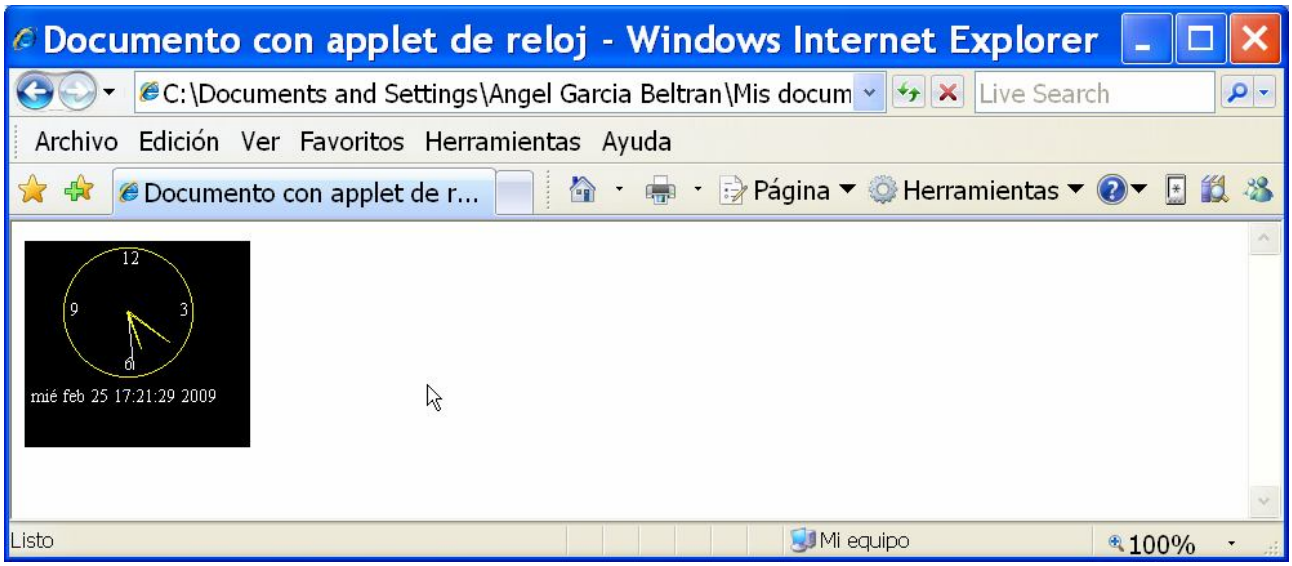

Figura 1.9. Visualización de un *applet* con parámetros

También puede emplearse el visor de *applets* del Kit de Desarrollo de Java (Figura 1.10). Desde la línea de comandos:

```
$:/>appletviewer ejemplo.html↵
```

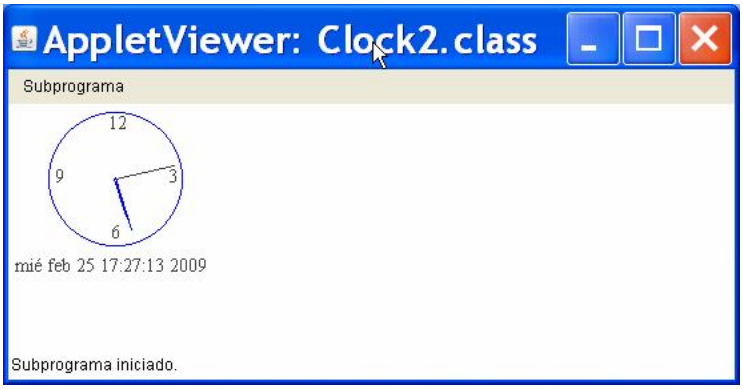

Figura 1.10. Ventana del visor de applets del Kit de Desarrollo de Java

Creación de un applet de Java

Las etapas para la creación de un *applet* de Java y su posterior inclusión en un documento HTML son las siguientes:

1. Crear un archivo fuente `MiApplet.java` con un editor de texto.

```
/** Este es un applet muy sencillo */
import java.applet.Applet;
import java.awt.Graphics;
public class MiApplet extends Applet {
   public void paint (Graphics g) {
```

```
        g.drawString("Hola, me llamo Angel", 10, 100);
    }
}
```

Las dos primeras líneas de código:

```
import java.applet.Applet;
import java.awt.Graphics;
```

hacen, respectivamente, que las clases `Applet` y `Graphics` estén disponibles para el resto del código siguiente. La tercera línea de código:

```
public class MiApplet extends Applet
```

Introduce una nueva clase llamada indicando que es una subclase de `Applet`. Las líneas de código restantes:

```
public void paint (Graphics g) {
    g.drawString("Hola, me llamo Angel", 10, 100);
}
```

declaran una operación (*método*) llamada `paint` que, a su vez, llama a otra denominada `drawString` que opera sobre un parámetro `g` de la clase `Graphics`.

2. Compilar el programa anterior para generar el archivo de *bytecodes* `MiApplet.class`.

3. Crear el siguiente documento HTML (`docu.html`) que incluye el applet anterior ya compilado (`MiApplet.class`). Para crear el documento HTML puede emplearse cualquier editor de texto (por ejemplo, el Block de Notas de Windows).

```
<html>
  <head>
    <title>Documento con applet muy sencillo</title>
  </head>
  <body>
    Este es el contenido del cuerpo.
    <applet code="MiApplet.class" height=”120” width=”150”>
    Si lees este texto, tu navegador no entiende Java
    </applet>
  </body>
</html>
```

4. Reconstrucción de documento HTML con un navegador (Figura 1.11).

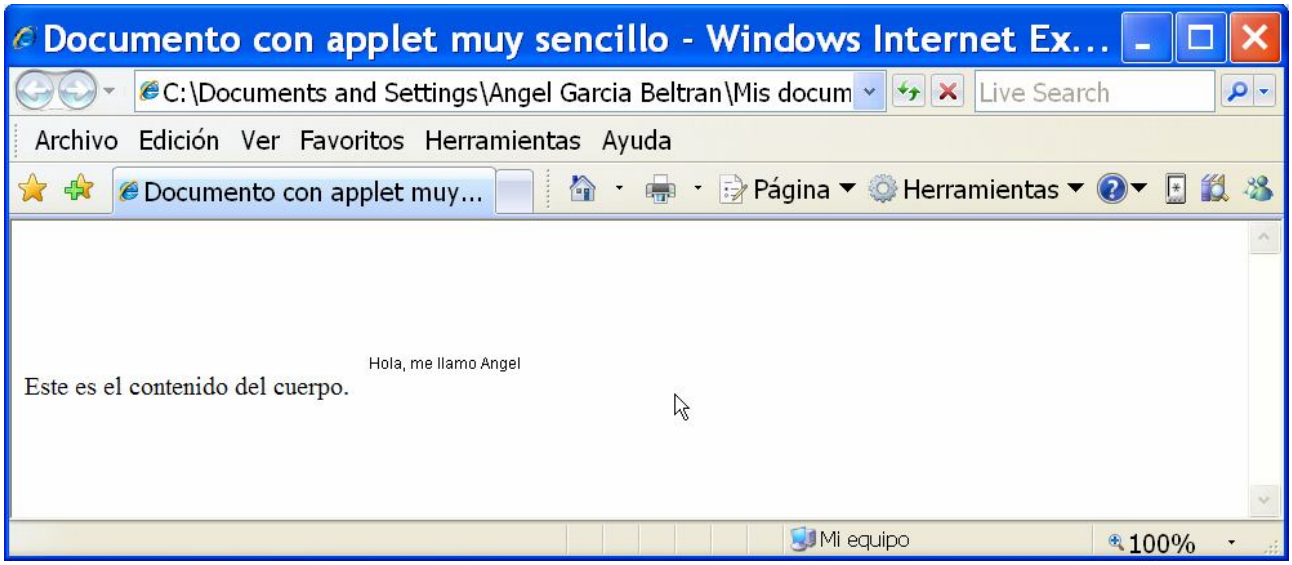

Figura 1.11. Visualización de un documento HTML con un *applet* de Java

5. También puede emplearse el visor de applets para visualizarlo. Desde la línea de comandos:

```
$:/>appletviewer docu.html↵
```

El resultado se muestra en la Figura 1.12.

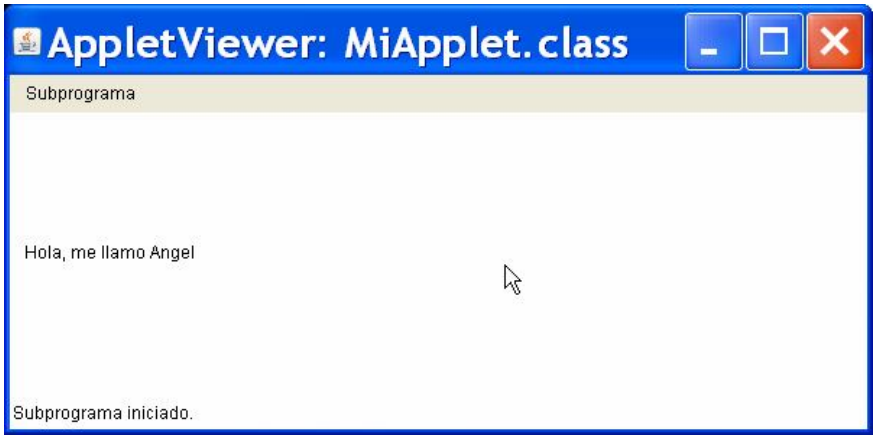

Figura 1.12. Ventana del visor de applets del Kit de Desarrollo de Java