

Tema 3

Introducción a Java

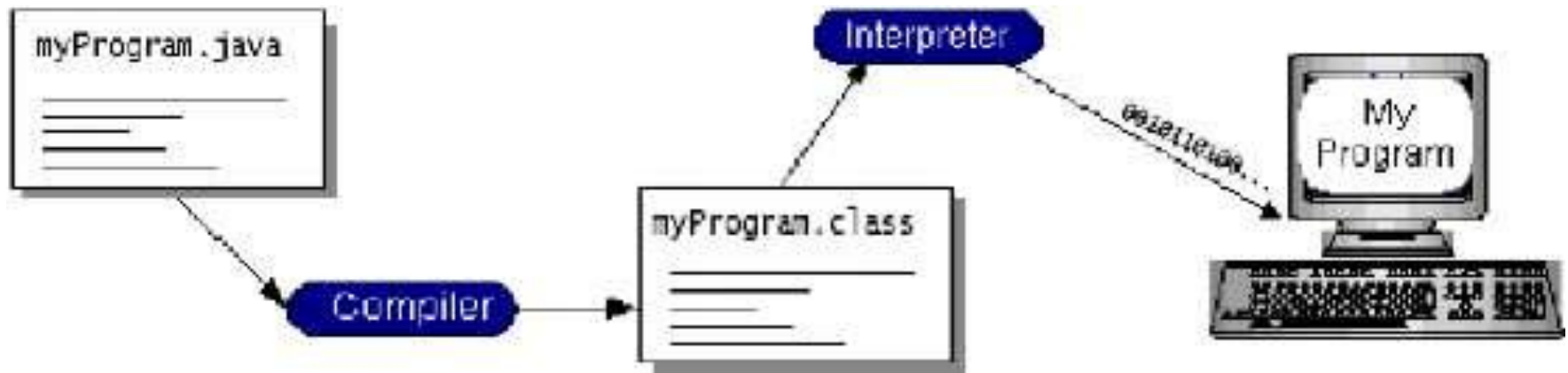


1. Introducción

- Fue creado por **Sun Microsystem**. Inicialmente fue diseñado para la programación de **microsistemas**, y fue difundido con una nueva orientación (**Internet**). Su sintaxis es muy similar a la de **C++**.
- Es un lenguaje **sencillo**. Los creadores de Java se basaron en C++ pero eliminaron la mayoría de sus complejidades, aunque Java posee una clase llamada String que reemplaza el tipo char[], y también tiene a Garbage Collector que es el comecocos de Java, elimina toda la memoria que no se utiliza. No soporta tipos de datos: struct, union y puntero. No soporta typedef ni #define.
- Java es un lenguaje para el desarrollo de software **Orientado a Objetos**, esta concebido para trabajar en un **entorno conectado en red** y cuenta con una amplia biblioteca de clases para comunicarse mediante TCP/IP.

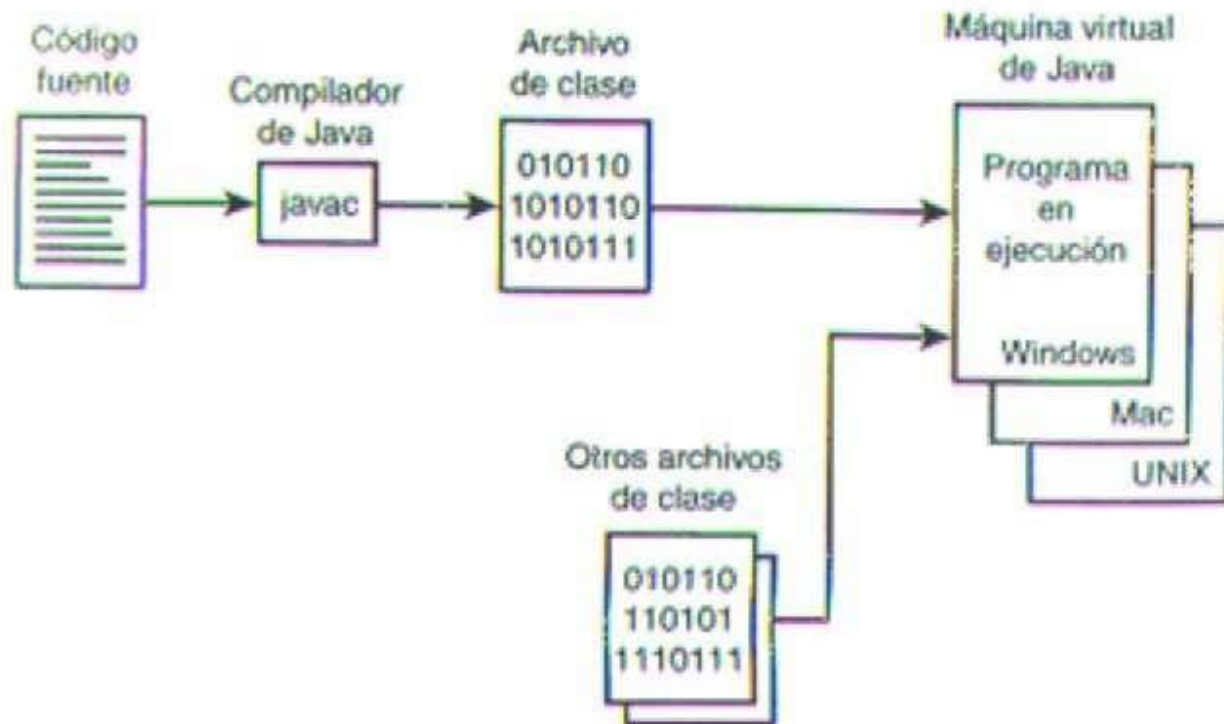
1. Introducción

- El compilador de Java traduce el código fuente a un código intermedio (**bytecode**), estos son **interpretados** en cualquier entorno donde exista un interprete de Java. Este interprete se le llama Máquina Virtual de Java o Java Virtual Machine (**JVM**).



1. Introducción

- Java es un lenguaje interpretado: Escribe el código una vez y ejecútalo en cualquier parte (write once, run anywhere).





1. Introducción

- Java no genera un ejecutable diferente cada vez que compilamos en una plataforma distinta sino que, independientemente de la plataforma, se generará el mismo código intermedio (bytecode), que después será ejecutado en cualquier plataforma por un intérprete específico para la plataforma.
- **Alto rendimiento.** Un lenguaje interpretado acostumbra a ser entre 20 y 100 veces más lento que el mismo programa compilado y ejecutado. Teniendo esto en cuenta, el rendimiento de Java es más que aceptable.
- Es un lenguaje **robusto** ya que no se interrumpe fácilmente a consecuencia de fallos, un lenguaje de estas características suele tener más restricciones a la hora de programar.
- Es un lenguaje **seguro**, ya que se critica mucho la seguridad, todos los navegadores poseen una 'sand box' (entorno de pruebas), existen tecnologías de firma digital para confiar en un determinado código Java y a su vez existen políticas de seguridad para controlar de una manera más precisa qué puede o no puede hacer.
- Es **portable** ya que los bytecodes son interpretados en cualquier plataforma donde exista una JVM. El uso de estándares (como UNICODE) permite obtener los mismos resultados en todas las plataformas. Los tipos de datos ocupan estrictamente lo mismo en cualquier plataforma.



1. Introducción

- Es **multithread** ya que soporta varias tareas a la vez, posee una serie de clases que facilitan su utilización “multihilo”.
- Es **dinámico**, el código C++, a menudo requiere una recompilación completa si cambia una clase, Java emplea un método de interfaces para evitar estas dependencias y recompilaciones.
- La plataforma Java es el entorno hardware y/o software donde se ejecuta un programa, esta es el **intérprete** de Java (la Máquina Virtual Java, JVM). El **API** Java (Interfaz de Programación de Aplicaciones) es un conjunto de clases ya desarrolladas que ofrecen un gran abanico de posibilidades al programador.



1. Introducción

- Trabajamos con el JDK (Java Development Kit) que no contiene ninguna herramienta gráfica para el desarrollo de programas pero contiene aplicaciones de consola y herramientas de compilación y depuración. El cuál incluye la máquina virtual JRE (Java Runtime Environment) que tiene los mínimos necesarios para ejecutar una aplicación.
- La información de la API no está incluida en el JDK y es necesario descargarla a parte. Esta documentación es la herramienta básica de consulta para todo desarrollador de Java por lo que es altamente recomendable que se descargue. Se puede descargar de la Web de Sun.

1. Introducción

- El código fuente de un programa Java se puede editar utilizando cualquier editor de textos (NotePad, Crimson, Vi, Emacs, ...) Por ejemplo:

```
public class HolaMundo
{
    /*lo único que hace este programa es mostrar la cadena Hola mundo! Por pantalla*/
    public static void main(String[] args)
    {
        System.out.println("Hola mundo!");
    }
}
```

- El **compilador javac** recibe el código fuente en un fichero con extensión .java y genera el archivo .class, que es el bytecode que recibe el entorno de ejecución. Para ejecutar un programa Java ya compilado, usaremos el intérprete Java. Por ejemplo:

- ☐ Fichero fuente: HolaMundo.java
- ☐ Creamos el bytecode: javac HolaMundo.java
- ☐ Lo ejecutamos: java HolaMundo

Para ejecutar el programa debemos tener el .class correspondiente y hemos de pasar el nombre de la clase principal, aquella donde está el método main, sin ningún tipo de extensión.



1. Introducción

- El entorno de JDK es de tipo línea de comando. Constituye todas las herramientas necesarias para desarrollar aplicaciones Java, así pues consta:
 - del compilador/linkador (javac),
 - del intérprete de aplicaciones (java),
 - de un debugger (jdb),
 - de un visor de applets (appletviewer),
 - de un programa de generación automática de documentación (javadoc), entre otros.
- Existen entornos de desarrollo como pueden ser NetBeans, Eclipse, Jbuilder, ... El J++ de Microsoft no es un producto Java ya que no era compatible con las especificaciones de Sun.
- El IDE de programación que vamos a utilizar es NetBeans versión 6.8.
 - <http://www.netbeans.org>
- Podemos descargar la paquetización que Oracle ofrece para NetBeans y el JDK en un único fichero



1. Introducción

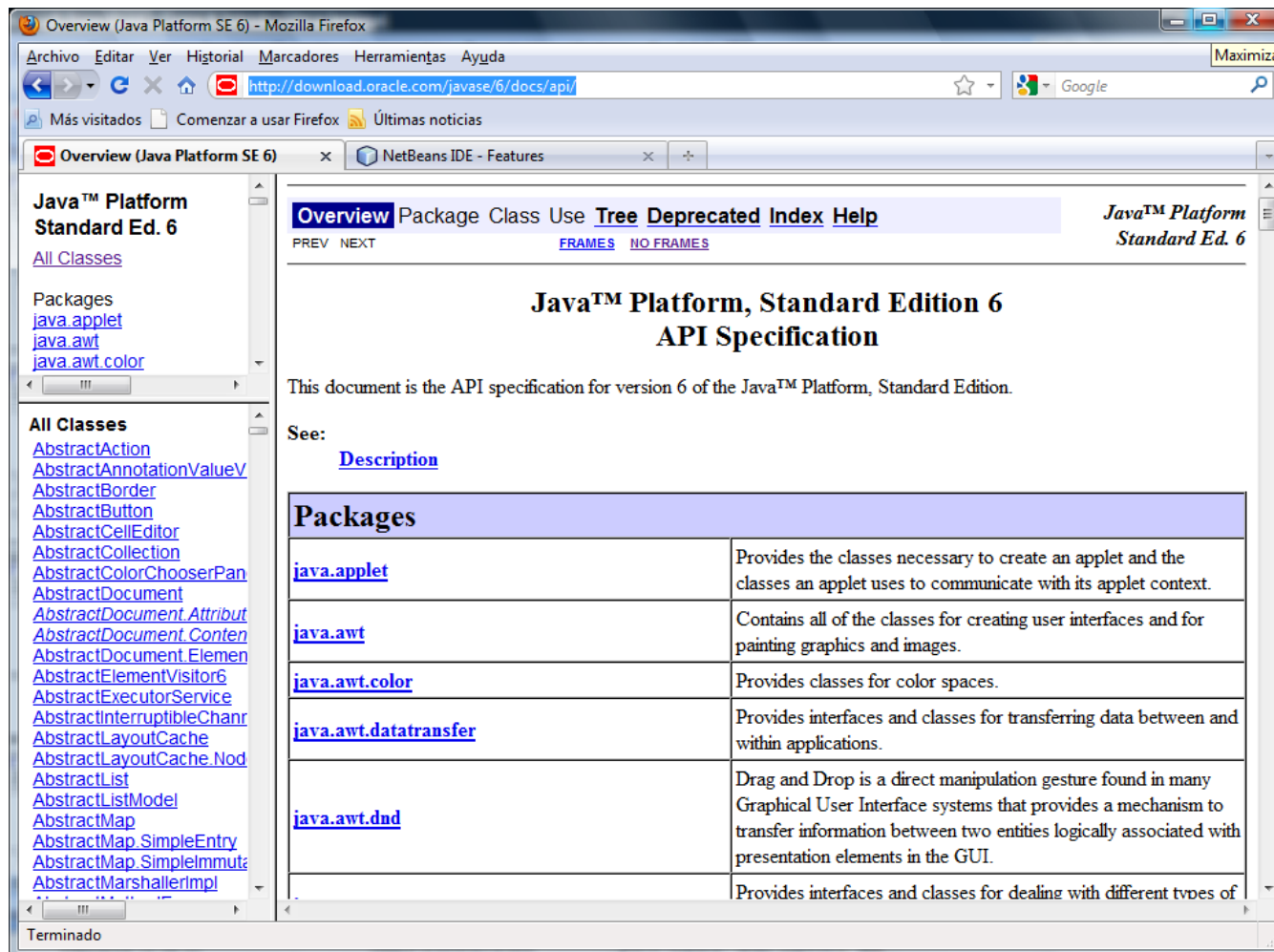
- Contenido del JDK

- ☐ /bin: las herramientas y utilidades del JDK.
- ☐ /lib: las librerías del JDK.
- ☐ /include: los archivos C/C++ utilizados para construir la JVM.
- ☐ /demo: una variedad de ejemplos escritos en Java.
- ☐ /jre: la JVM sin herramientas de desarrollo.
- ☐ /src.zip: el código fuente de las APIs comprimido.

1. Introducción

- La documentación

- <http://download.oracle.com/javase/6/docs/api/>



Overview (Java Platform SE 6) - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Ayuda

<http://download.oracle.com/javase/6/docs/api/>

Más visitados Comenzar a usar Firefox Últimas noticias

Overview (Java Platform SE 6) NetBeans IDE - Features

Java™ Platform Standard Ed. 6

[All Classes](#)

Packages

- [java.applet](#)
- [java.awt](#)
- [java.awt.color](#)

All Classes

- [AbstractAction](#)
- [AbstractAnnotationValueV](#)
- [AbstractBorder](#)
- [AbstractButton](#)
- [AbstractCellEditor](#)
- [AbstractCollection](#)
- [AbstractColorChooserPan](#)
- [AbstractDocument](#)
- [AbstractDocument.Attribut](#)
- [AbstractDocument.Conten](#)
- [AbstractDocument.Elemen](#)
- [AbstractElementVisitor6](#)
- [AbstractExecutorService](#)
- [AbstractInterruptibleChanr](#)
- [AbstractLayoutCache](#)
- [AbstractLayoutCache.Nod](#)
- [AbstractList](#)
- [AbstractListModel](#)
- [AbstractMap](#)
- [AbstractMap.SimpleEntry](#)
- [AbstractMap.SimpleImmuts](#)
- [AbstractMarshallerImpl](#)

Overview Package Class Use Tree Deprecated Index Help

PREV NEXT

[FRAMES](#) [NO FRAMES](#)

Java™ Platform, Standard Edition 6

API Specification

This document is the API specification for version 6 of the Java™ Platform, Standard Edition.

See: [Description](#)

Packages

java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
	Provides interfaces and classes for dealing with different types of

Terminado