

# Bienvenidos

## Java Programming: A Comprehensive Hands-On

**David R. Luna G.**  
**davidrlunag@gmail.com**  
**0412-3111011**





**PRESENTACION**

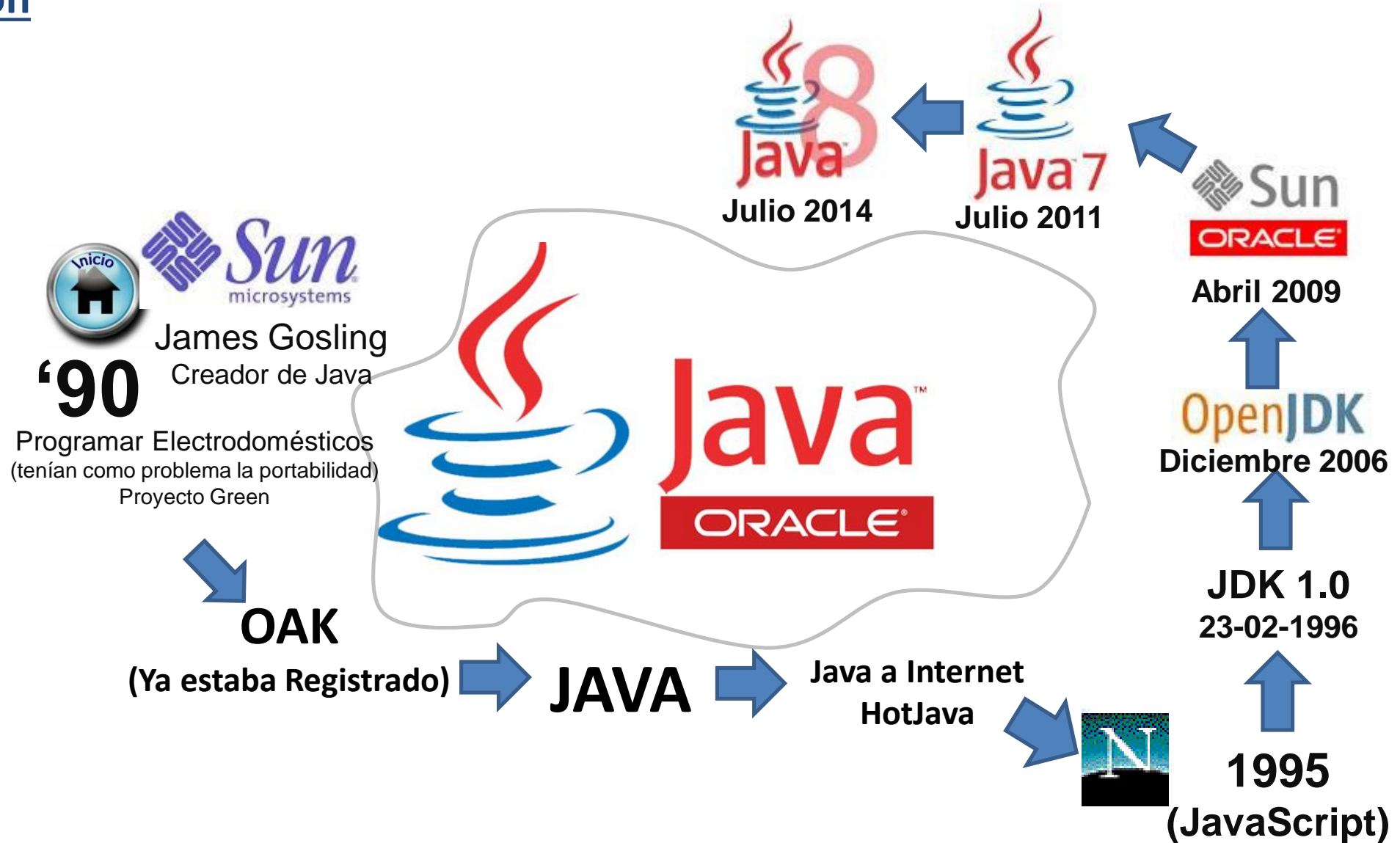
# Objetivos del Curso

- ✓ Construir aplicaciones de consola y con GUI usando Programación Orientada a Objetos
- ✓ Crear aplicaciones usando las Librerías de clases de Java
- ✓ Desarrollar Interfaces de Usuario Independientes de la Plataforma
- ✓ Leer y Escribir data en archivos usando Java Streams
- ✓ Manejo de Data desde base de datos relacionales con JDBC

# Temario del Curso

- ✓ **Introducción al Lenguaje Java**
- ✓ Estructura del Lenguaje Java
- ✓ Herramientas de Desarrollo Java
- ✓ Programación Orientada a Objetos con Java
- ✓ Manejo de Datos con Archivos
- ✓ Trabajando con Base de Datos Relaciones
- ✓ Desarrollando GUIs

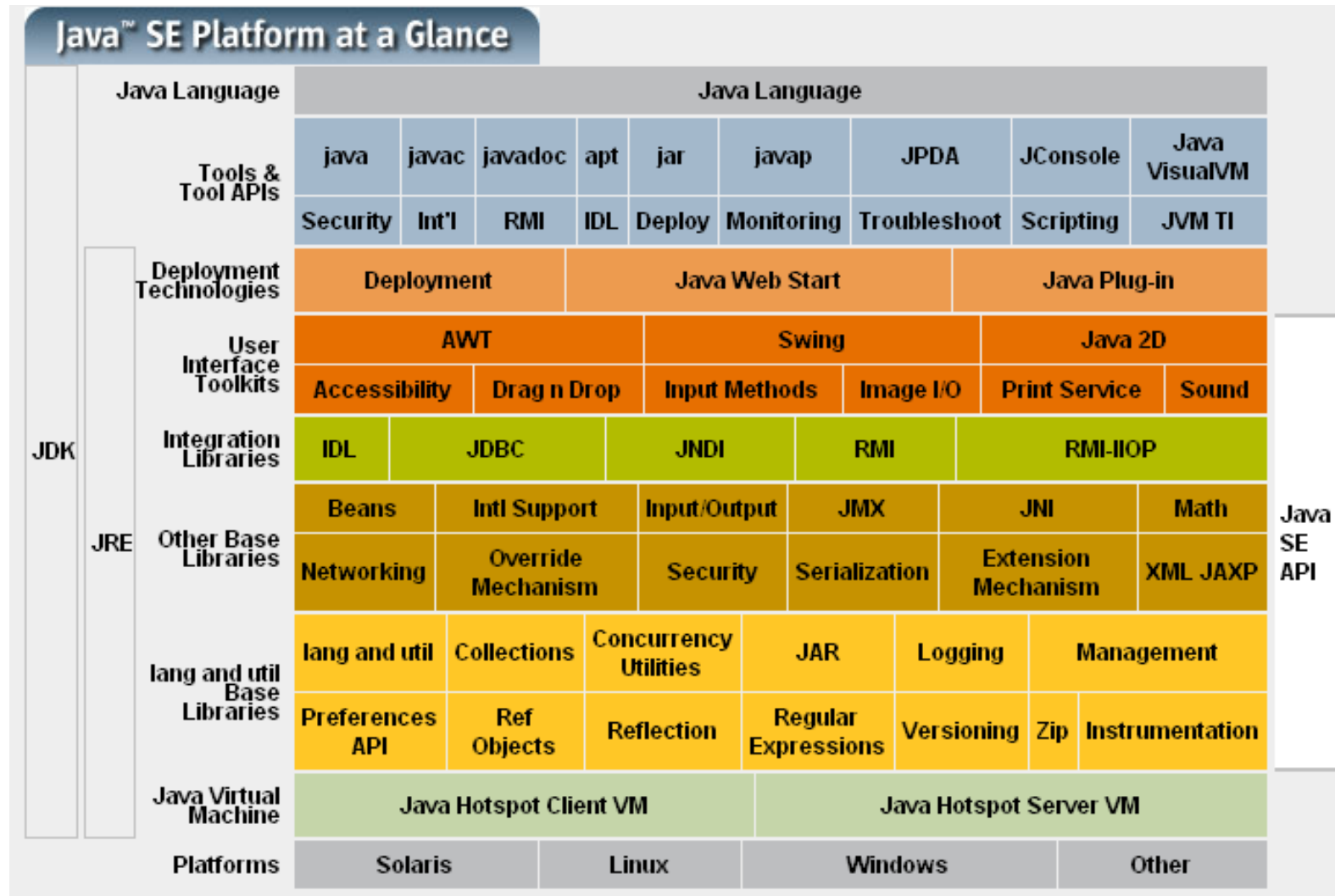
## Introducción



## Introducción



# Introducción



# Introducción

## Datos de Interés

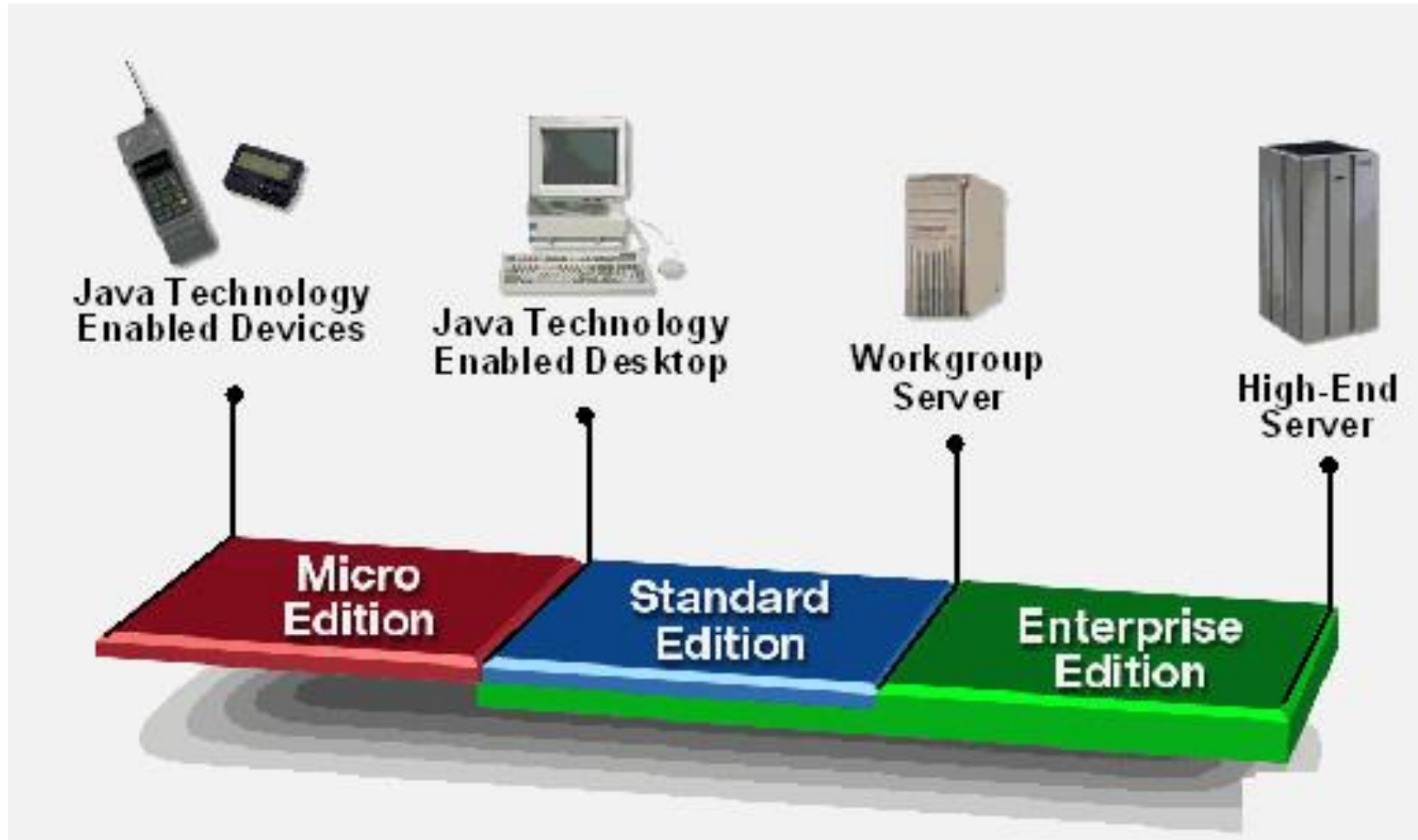
1. Más de 5 millones de downloads del SDK hasta la fecha.
2. Más de 10 mil downloads diarios.
3. 50+ proveedores de Application Servers.
4. Más de 3 millones de desarrolladores certificados.
5. Java está disponible para Windows, Solaris, Linux, MacOS X, HP-UX, OpenVMS, Tru64 Unix, IRIX, AIX, AS/400, OS/2, OS390...
6. J2SE es la capa fundamental en la que están basados J2EE y los servicios web Java.
7. La plataforma Java ha dado pasos muy grandes en cuanto a performance, escalabilidad y funcionalidad.
8. Java 1.7 es completamente compatible con versiones anteriores.
- 9.- En más de 4.500 millones de dispositivos, en más de 900 millones de ordenadores, en más de 2.000 millones de teléfonos y en más de 1500 millones de tarjetas inteligentes.





## Introducción

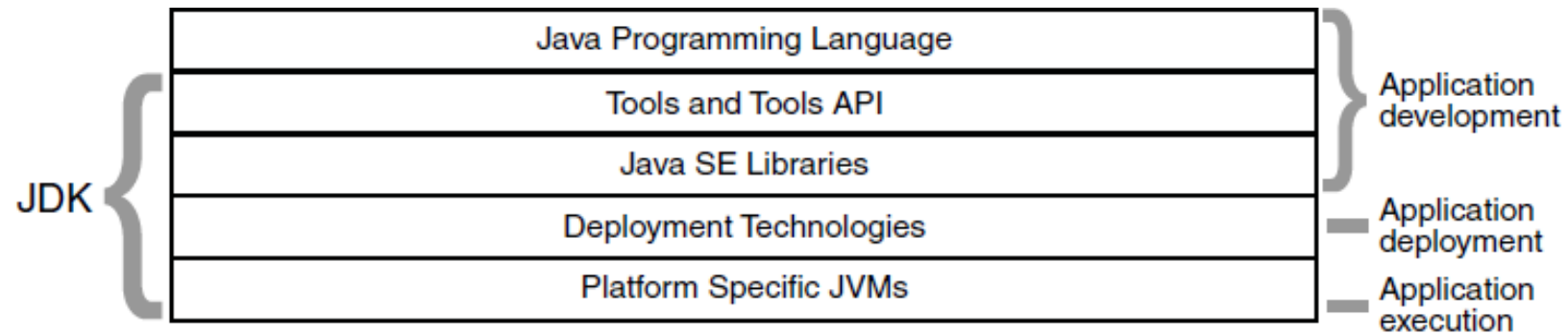
### Los 3 Ambientes



# Introducción

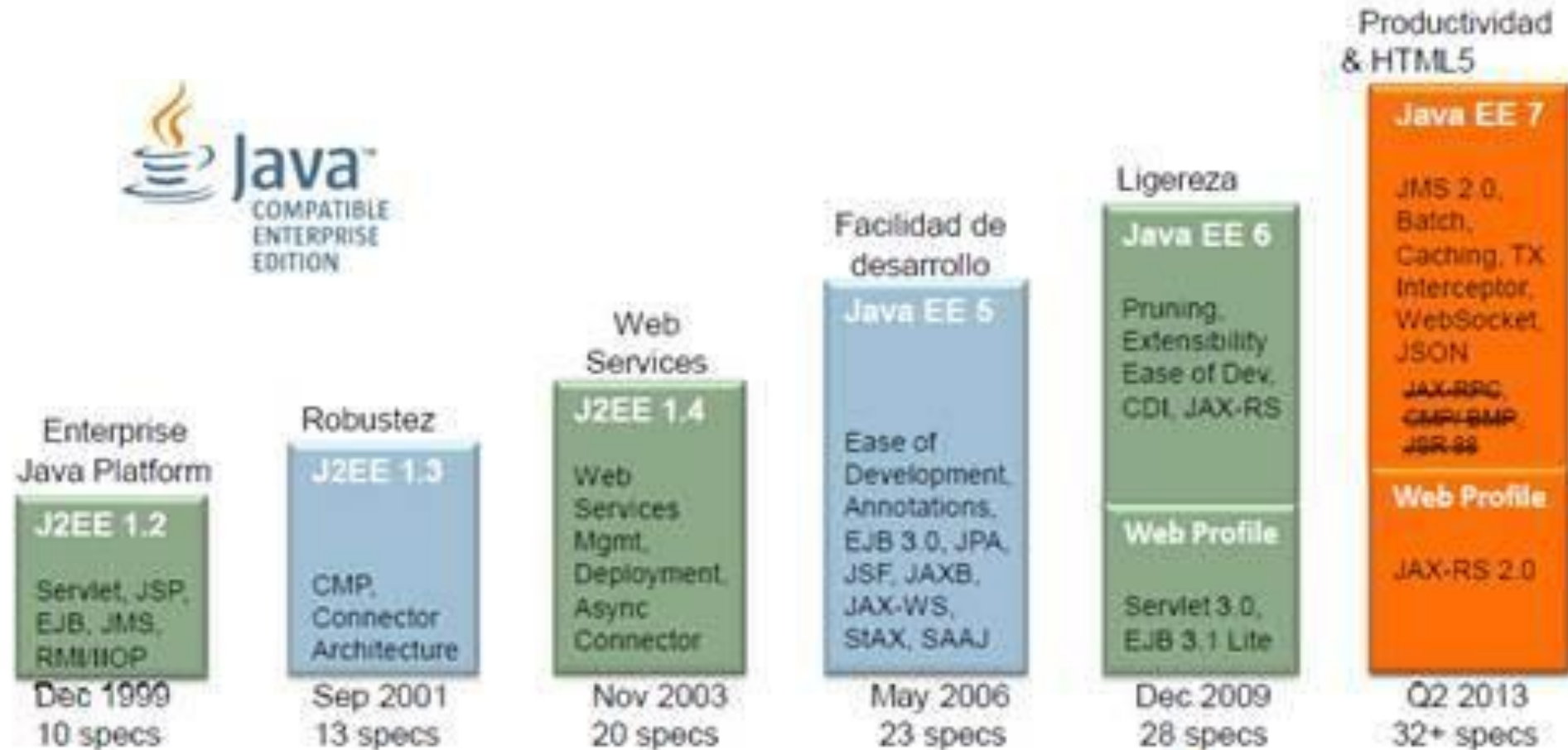
The JDK contains components to perform the following tasks:

- Develop Java technology applications
- Deploy Java technology applications
- Execute Java technology applications



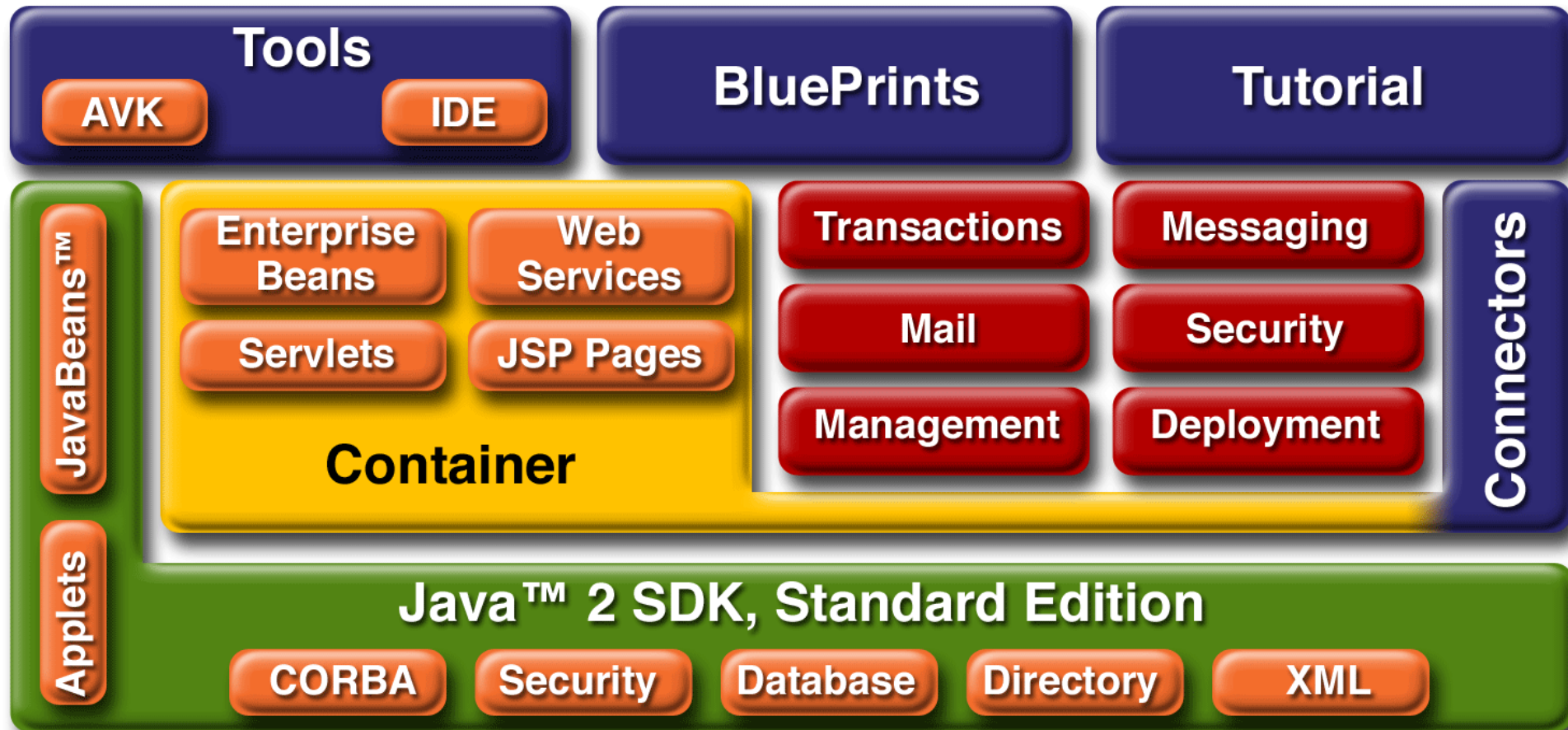
Introducción

# Java Platforms

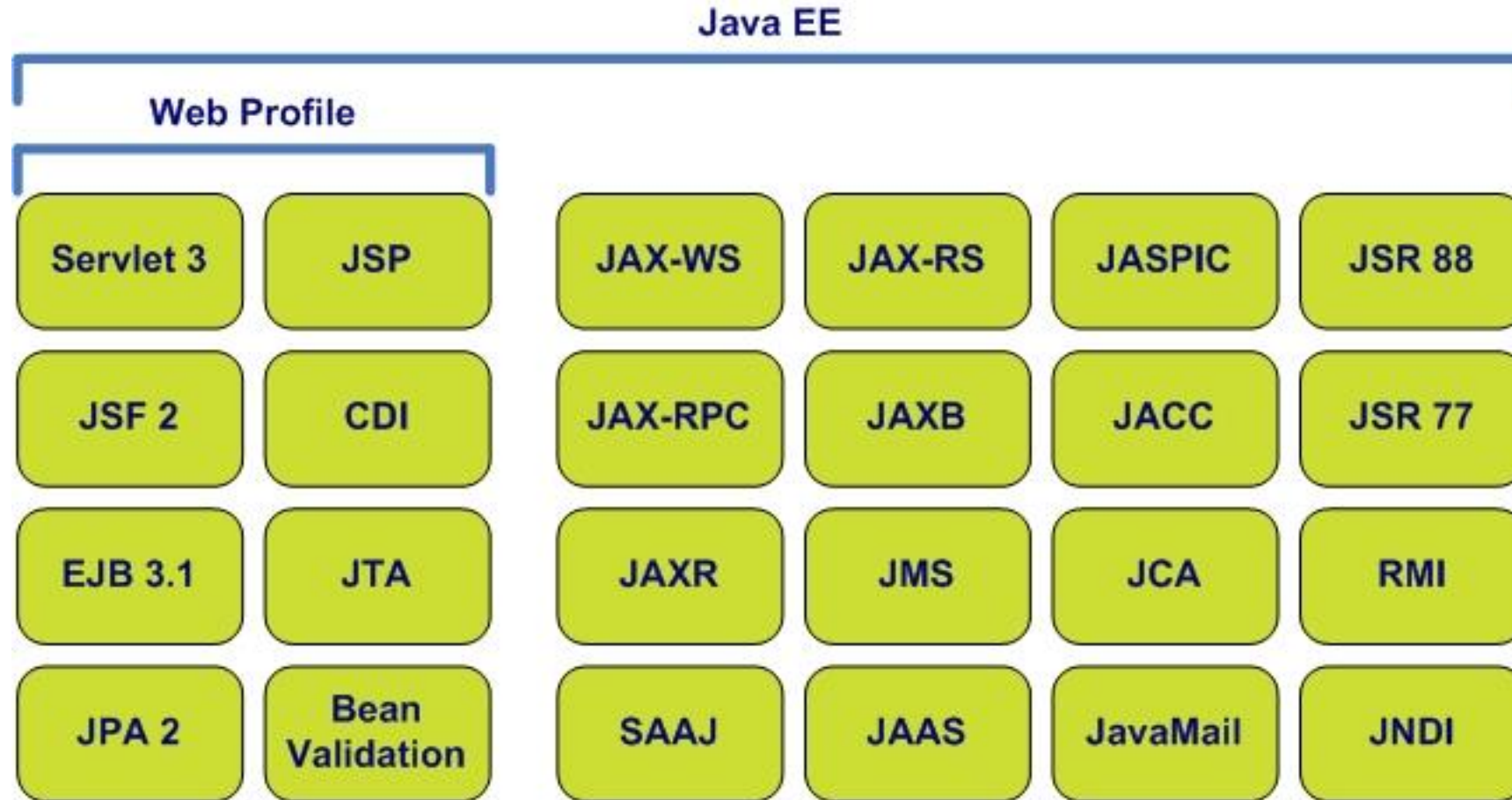


Introducción

# Java Platforms



# Java Platforms





## Introducción

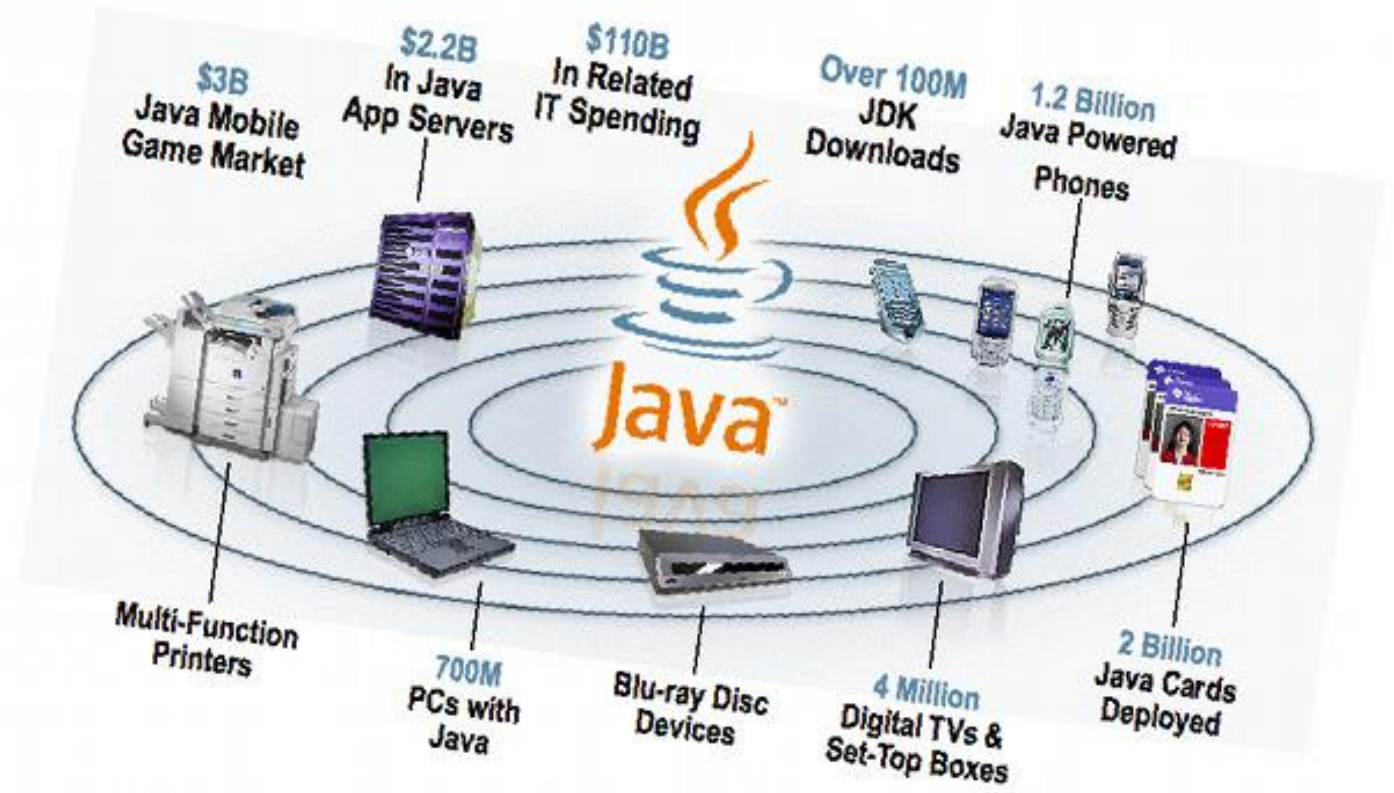
# Java Overview



## Introducción



## Introducción





## Introducción

### Un Lenguaje de Programación

1. Java es un lenguaje de tercera generación (3GL).
2. Es similar al lenguaje C.
3. Tiene mucho de la sintaxis del lenguaje C, sin ser C o C++, ni mucho menos un superconjunto de C.
4. A diferencia de otros lenguajes, Java permite escribir programas especiales llamados Applets que se ejecutan en un browser de manera segura.
5. Un Applet no puede escribir en disco o memoria sin previa autorización, por lo que la seguridad está garantizada. Por lo tanto no se pueden introducir virus.

### Plataforma para el Desarrollo de Aplicaciones

1. Una plataforma es un ambiente de software o hardware sobre el que se ejecuta un programa.
2. La plataforma Java es sólo una plataforma de software, que se ejecuta por encima de otras plataformas de software (Sistemas Operativos).
3. Permite desarrollar aplicaciones de manera independiente a la plataforma.
4. El compilador de Java no produce código ejecutable nativo, genera byte code.
5. Un byte code es un formato especial escrito en hexadecimal, byte a byte como lo siguiente:  
**CA FE BA 00 B5 01 20 E5 2D 03 08 14 C2 AA 4A 3C 32** Cada byte code es exactamente el mismo sobre cada plataforma.

Un byte code se compila y requiere ser interpretado para ejecutarse.



## Introducción

**La plataforma Java tiene dos componentes principales:**

### ***La Java Virtual Machina (JVM) o Maquina Virtual de Java***

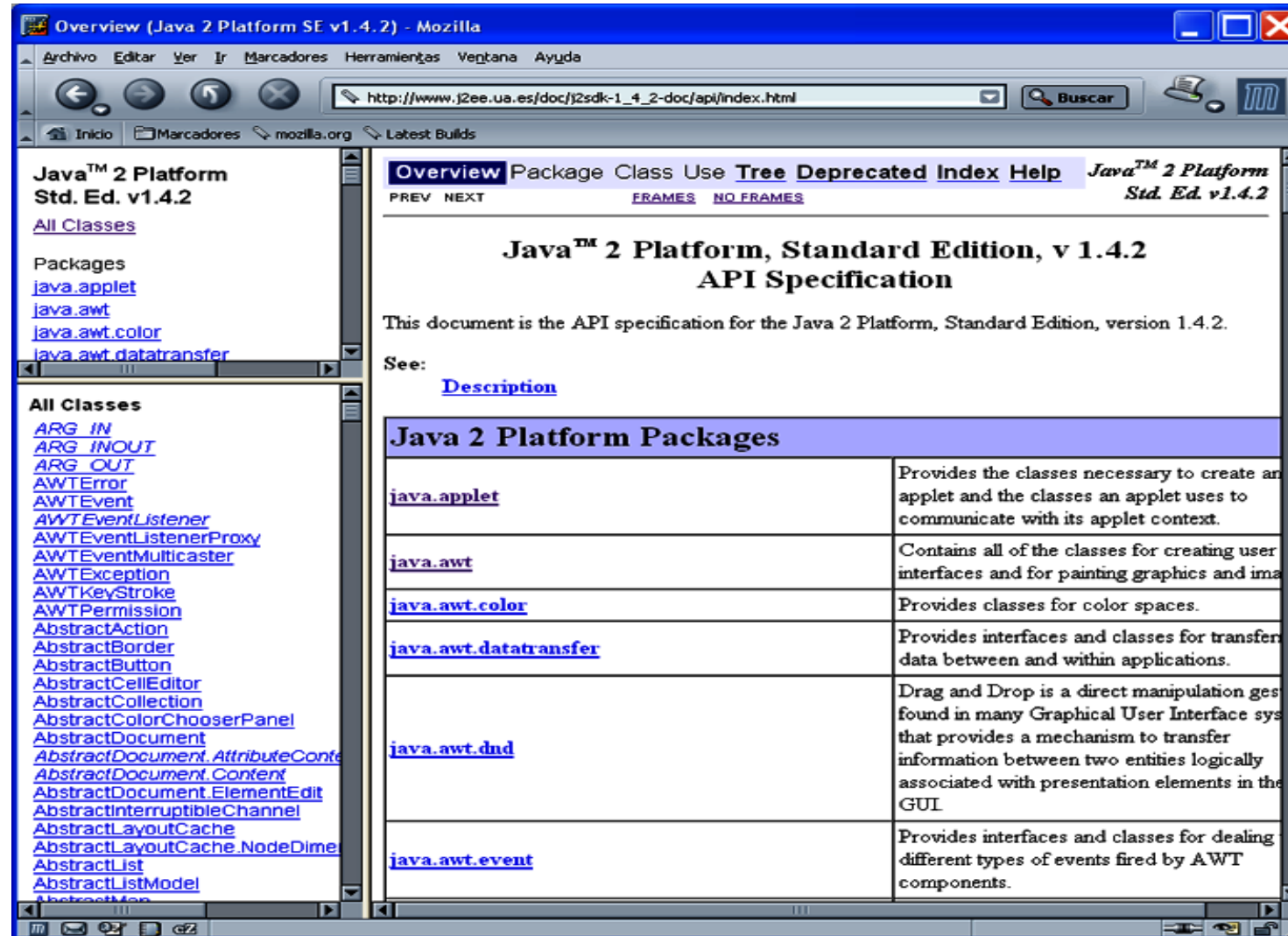
Es la base de la plataforma Java y puede ser incorporada en la mayoría de las plataformas basadas en hardware y sistemas operativos. Contiene el interprete de Java.

### ***La Java Application Programming Interface - Java API***

Es una colección de componentes de software que proveen una amplia gama de funcionalidades, como GUIs, I/O, etc. Está agrupada en paquetes o librerías de componentes relacionadas.



# Introducción



The screenshot shows a Mozilla browser window titled "Overview (Java 2 Platform SE v1.4.2) - Mozilla". The address bar displays the URL [http://www.j2ee.ua.es/doc/j2sdk-1\\_4\\_2-doc/api/index.html](http://www.j2ee.ua.es/doc/j2sdk-1_4_2-doc/api/index.html). The page content is the "Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification". It includes a navigation menu on the left with links to "All Classes" and "Packages". The main content area features a table titled "Java 2 Platform Packages" with the following entries:

Java 2 Platform Packages	
<a href="#">java.applet</a>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<a href="#">java.awt</a>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<a href="#">java.awt.color</a>	Provides classes for color spaces.
<a href="#">java.awt.datatransfer</a>	Provides interfaces and classes for transferring data between and within applications.
<a href="#">java.awt.dnd</a>	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.
<a href="#">java.awt.event</a>	Provides interfaces and classes for dealing with different types of events fired by AWT components.

<https://docs.oracle.com/javase/8/docs/api/>



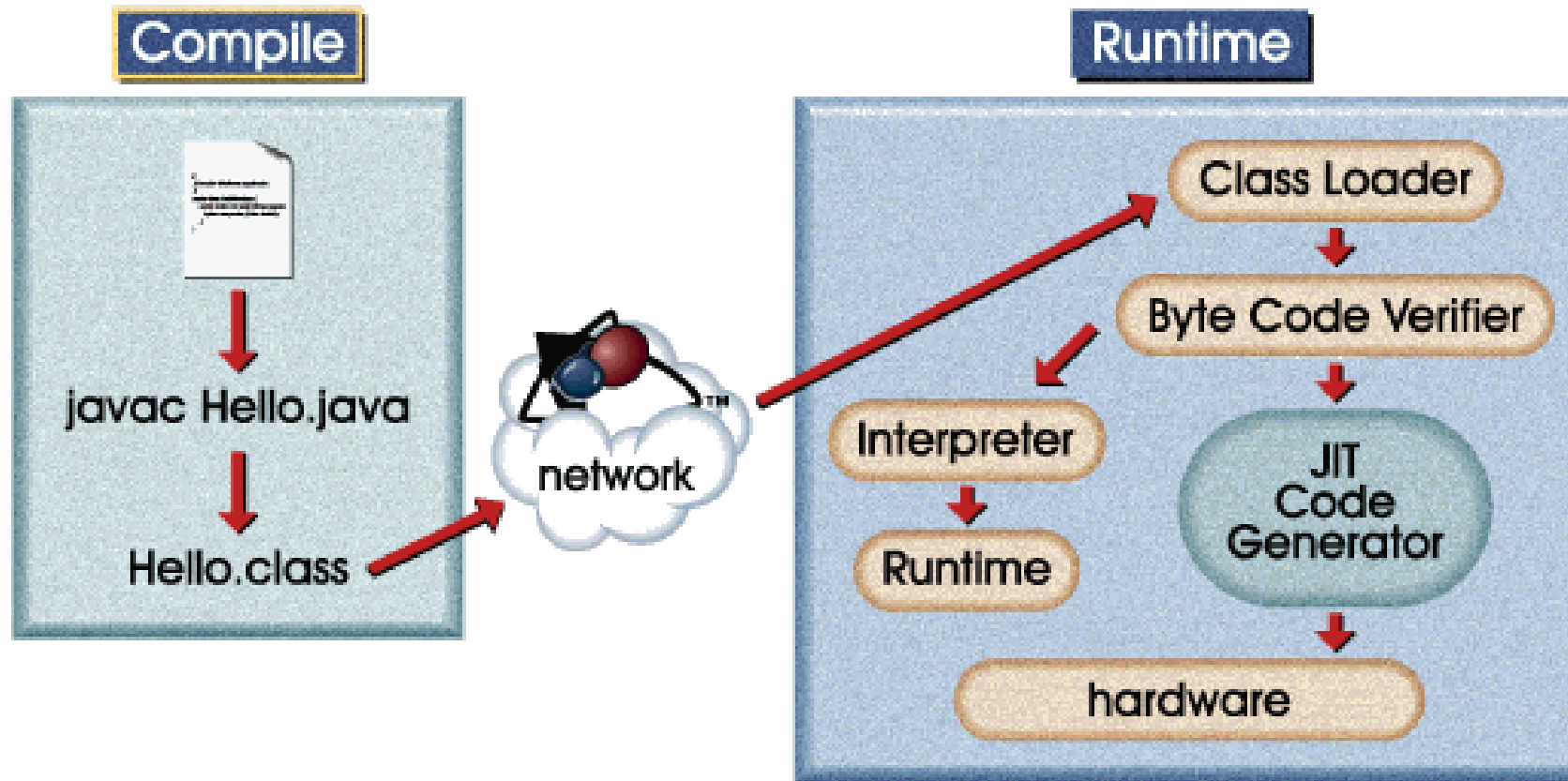
## Introducción

### **Básicamente, el JDK consiste de:**

- el compilador Java, javac
  - javac nombre\_archivo.java
  - Produce un nombre\_archivo.class (ByteCode)
- el intérprete Java, java
  - Java nombre\_archivo (compilado)
  - Lo ejecuta la JVM
  - Busca el Metodo main()
- un visualizador de applets, appletviewer
- el debugger Java, jdb (que para trabajar necesita conectarse al server de Sun)
- el generador de documentación, javadoc



## Introducción



## Introducción

### Estructura de un Programa en Java

[declaracion\_de\_paquetes]

[inclusion\_de\_paquetes]

declaración\_de\_la\_clase

```
package prueba;  
import java.net.*;  
public class prueba{ .... }
```

### Antes de Comenzar

- .- Es importante destacar, que el nombre de la clase, debe ser el mismo del archivo fuente, esto es una regla del compilador de java.
- .- En java todas las instrucciones terminan con punto y coma (;)
- .- Java es Case Sensitive, es decir, diferencia entre minúscula y mayúsculas.
- .- Para indicar inicio se utiliza { y para indicar el fin se utiliza }.
- .- Utiliza caracteres UNICODE



## Introducción

### Manejo de Paquetes

Un paquete nos permite agrupar un número de clases relacionadas, brindando protección de acceso y administración del espacio de nombres en la aplicación.

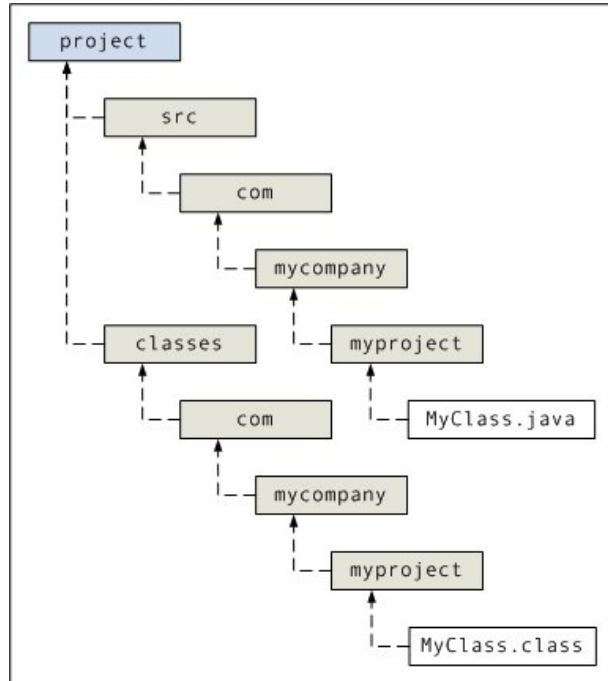
Sintaxis básica:

```
package <top_pkg_name>[.<sub_pkg_name>]*;
```



## Introducción

### Manejo de Paquetes



Los paquetes son escritos en minúsculas para evitar conflictos con los nombres de clases o interfaces.

Las empresas usan su dominio inverso para iniciar sus paquetes, por ejemplo, `com.example.mypackage`

Los paquetes son la forma de organizar archivos en Java, son usados cuando un proyecto consiste de múltiples módulos. También ayudan a resolver conflictos de nombre y permite proteger la data al no permitir ser utilizadas por clases no autorizadas.





## Introducción

### Manejo de Paquetes

Para importar un importar una clase en nuestro programa, debemos utilizar la sentencia import.

```
import curso.programming.sintaxis.Car;
```

Para importar todas las clases contenidas en un paquete en particular, se utiliza el \* como comodín.

```
import graphics.*;
```



## Introducción

### Manejo de Paquetes

Importando `java.awt.*` importa todos los tipos contenidos en el paquete `java.awt`, pero no importará `java.awt.color`, `java.awt.font`, o cualquier otro paquete dentro de `java.awt`. Por lo que si necesitas usar clases de `java.awt` y de `java.awt.color` se debe hacer así:

```
import java.awt.*;  
import java.awt.color.*;
```



## Introducción

### Manejo de Paquetes

Si una variable de un paquete comparte el nombre con otra clase de otro paquete, se debe referir a cada tipo por su nombre calificado:

```
curso.programming.sintaxis.Car car;
```



## Introducción

### El API de Java

**java.lang:** contiene las clases esenciales como números, strings, objetos, compilador, run-time, seguridad y threads (es el único paquete que se incluye automáticamente en todo programa Java)

**java.io:** contiene las clases que manejan la Entrada/Salida, Serialización de objetos.

**java.util:** contiene clases útiles, que permiten manejar estructuras de datos, fechas, hora, strings, excepciones, etc.

**java.net:** contiene clases como URL, TCP, UDP, IP, etc. que permiten implementar aplicaciones distribuídas. Provee soporte para sockets.

**java.awt:** contiene clases para el manejo de la GUI, pintar gráficos e imágenes.

**java.awt.image:** contiene las clases para el manejo de imágenes.

**java.applet:** contiene clases útiles para la creación y manipulación de Applets y recursos para reproducción de audio.

**java.rmi:** contiene clases para soporte trabajar con objetos remotos.

**java.sql:** contiene clases para el manejo de base de datos relaciones (JDBC, JDBC-ODBC).

**java.security:** contiene clases e interfaces para manejar seguridad (criptografía, firmas digitales, encriptación y autenticación).



## Introducción

### Las Clases en Java

Una *clase* es una agrupación de *datos* (variables o campos) y de *funciones* (métodos) que operan sobre esos datos. A estos datos y funciones pertenecientes a una clase se les denomina *variables y métodos o funciones miembro*. Un programa se construye a partir de un conjunto de clases. Y todo en Java es una clase.

### Declarando una clase en Java

```
[< modificadores>] class <nombre_clase>
{
    [< declaración_atributos>]
    [< declaración_constructores>]
    [<declaración_metodos>]
}
```

Una vez definida e implementada una clase, es posible declarar elementos de esta clase de modo similar a como se declaran las variables del lenguaje (de los tipos primitivos *int*, *double*, *String*, ...). Los elementos declarados de una clase se denominan *objetos* de la clase. De una única clase se pueden declarar o crear numerosos *objetos*.



## Introducción

### El Método Main

El método *main()* es usado como punto de entrada para los programas de aplicación Java. Todos los programas deben poseer un método *main()* o éstos no podrán ser ejecutados. El método *main()* es el método de una clase que es ejecutado para iniciar un programa.

Las clases deben incluir el metodo main, para indicarle al compilador que son ejecutables (no es obligatorio), dentro de este método irá toda la secuencia de ejecución de nuestra aplicación.

### Definición del Método Main

```
public static void main(String args[]){  
    // Sentencias  
}
```



## Introducción

### Ejemplo de una Clase

```
public class Vehiculo {  
    private double velocidad;  
    public void setVelocidad(double value) {  
        velocidad = value;  
    }  
}
```

### Modificadores de Clase

**public:** Indica que la clase podrá ser accesible desde cualquier otra clase, ya que es pública.

**abstract:** Es un modificador especial que indica que la clase no puede instanciarse (no se pueden crear objetos a partir de ella).

**final:** Indica que la clase no puede utilizarse para crear subclases a partir de ella (no se puede heredar)



## Introducción

### Imprimiendo por Pantalla

**`System.out.println("mensaje a mostrar");`**

La misma pertenece al paquete `java.lang`, por lo que la podemos usar en cualquier programa sin hacer nada.

También podemos mandar a mostrar el valor de una variable:

**`System.out.println(variable);`**

Y podemos mostrar un mensaje, con una variable de la siguiente manera:

**`System.out.println("mensaje" + variable);`**

En este caso, el operador `+` concatena al mensaje el valor de la variable, en otras palabras, transforma las variables a `String` para poderlos mostrar por pantalla.





## Introducción

### Nuestro Primer Programa

```
public class Prueba{  
    public static void main(String args[]){  
        System.out.println("Hola");  
        System.out.println("Este es mi Primer Programa");  
    }  
}
```

**No olvides que el Archivo fuente debe llamarse igual que la clase**

Compilando

```
javac Prueba.java
```

Ejecutando

```
java Prueba
```

Resultado

```
Hola
```

```
Este es mi Primer Programa
```

