# *Introducción a la programación*

# *Java*

**La plataforma de programación Java**

- Historia
- La máquina virtual Java
- Herramientas de programación en Java

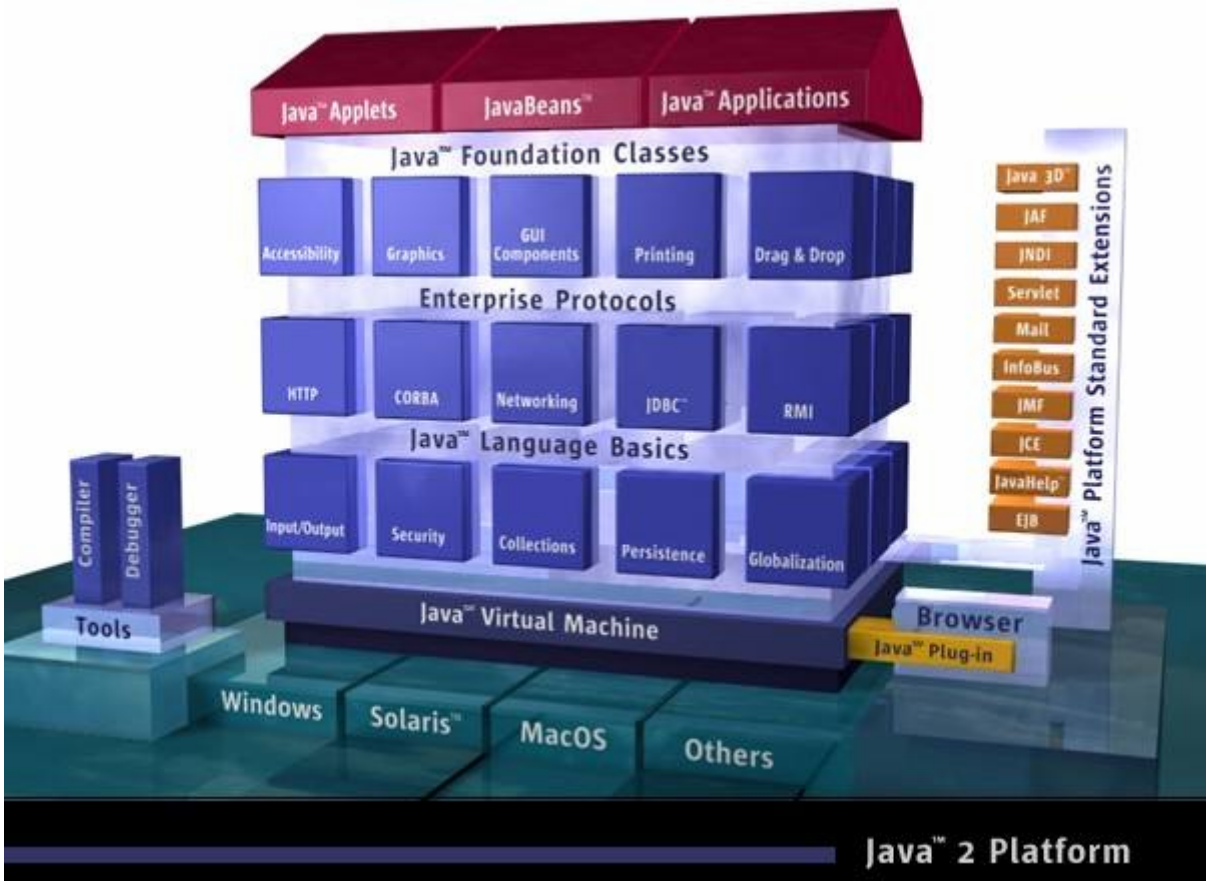
**Aplicaciones y applets**

- Aplicación de ejemplo
- Applet de ejemplo
- Fases en la creación y ejecución de programas en Java

**Características clave de Java**

- Mitos y realidades

# *La plataforma Java*




- La **máquina virtual Java**
  **(JVM: Java Virtual Machine)**

  Imprescindible para poder ejecutar aplicaciones Java.

- Las **bibliotecas** estándar de Java
  **(Java Application Programming Interface = Java API)**

  Amplia colección de componentes.

- El **lenguaje de programación Java**

  Para escribir aplicaciones.

# *Historia de Java*

*Hay versiones distintas sobre el origen, concepción y desarrollo de Java, desde la que dice que éste fue un proyecto que estuvo durante mucho tiempo por distintos departamentos de Sun sin que nadie le prestara atención hasta la más difundida, que presenta a Java como un lenguaje pensado para pequeños electrodomésticos:*

Hace algunos años, Sun Microsystems decidió intentar introducirse en el mercado de la electrónica de consumo y desarrollar programas para pequeños dispositivos electrónicos. Sun decidió crear una filial, denominada FirstPerson Inc..

El mercado inicialmente previsto para los programas de FirstPerson eran los equipos domésticos: microondas, tostadoras y, fundamentalmente, televisores interactivos. En este mercado, dada la falta de pericia de los usuarios, se requerían unos interfaces mucho más cómodos e intuitivos que los sistemas de ventanas del momento.

James Gosling decidió que las ventajas aportadas por la eficiencia de C++ no compensaban el gran coste de la prueba y depuración de aplicaciones C++. Gosling había estado trabajando en un lenguaje de programación que él había llamado **Oak**, el cual, aún partiendo de la sintaxis de C++, intentaba remediar las deficiencias que iba observando.

El primer proyecto en que se aplicó este lenguaje recibió el nombre de proyecto Green y consistía en un sistema de control completo de los aparatos electrónicos y el entorno de un hogar.

Para ello se construyó un ordenador experimental denominado ***7** (Star Seven). El sistema presentaba una interfaz basada en la representación de la casa de forma animada y el control se llevaba a cabo mediante una pantalla sensible al tacto. En el sistema aparecía Duke, la mascota de Java.

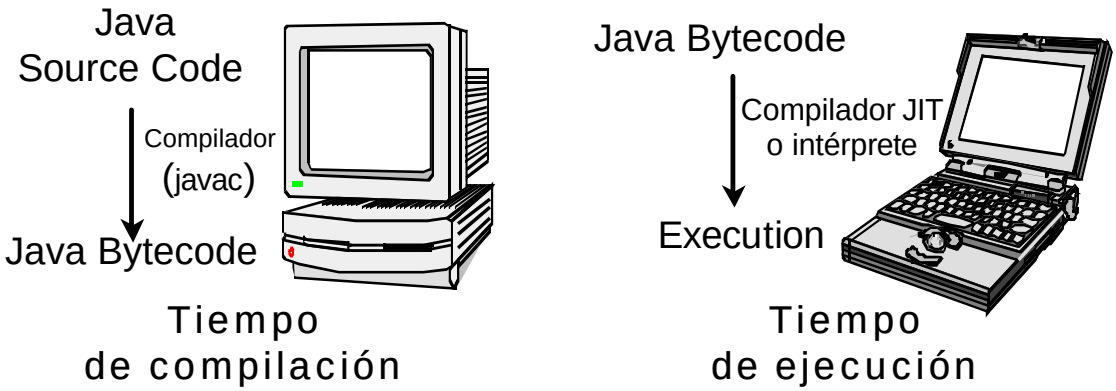
Posteriormente, se aplicó a otro proyecto de VoD (Video On Demand) en el que se empleaba como interfaz para la televisión interactiva. Ninguno de estos proyectos se convirtió nunca en un sistema comercial.

Cuando en Sun se dieron cuenta de que a corto plazo la televisión interactiva no iba a ser un gran éxito, urgieron a FirstPerson a desarrollar con rapidez nuevas estrategias que produjeran beneficios. No lo consiguieron y FirstPerson cerró en la primavera de 1994.

A pesar de este fracaso, Bill Joy, cofundador de Sun y uno de los desarrolladores principales del Unix de Berkeley, juzgó que Internet podía llegar a ser el terreno adecuado para disputar a Microsoft su primacía casi absoluta en el terreno del software y vio en Oak el instrumento idóneo para llevar a cabo estos planes. Tras un cambio de nombre, al estar Oak ya registrado como marca, el lenguaje Java fue presentado en sociedad en mayo de 1995 (Sun World'95).

http://java.sun.com/features/1998/05/birthday.html

## *La máquina virtual Java*




- El **compilador de Java** genera un código intermedio independiente de la plataforma (bytecodes).

- Los **bytecodes** pueden considerarse como el lenguaje máquina de una máquina virtual, la Máquina Virtual Java (JVM).

- Cuando queremos **ejecutar una aplicación Java**, al cargar el programa en memoria, podemos

  a) Interpretar los bytecodes instrucción por instrucción

  b) Compilar los bytecodes para obtener el código máquina necesario para ejecutar la aplicación en el ordenador (compilador JIT [Just In Time]).

De esta forma, podemos ejecutar un programa escrito en Java sobre distintos sistemas operativos (Windows, Solares, Linux...) sin tener que recompilarlo, como sucedería con programas escritos en lenguajes como C.

*Uso típico de Java*

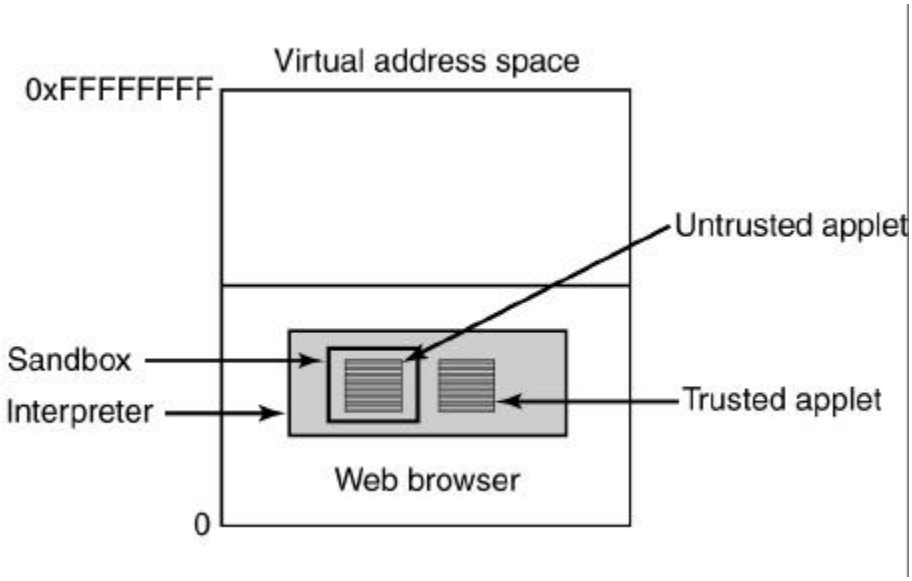
Distribución de aplicaciones a través de Internet

- Aplicaciones (programas independientes)
- Applets ("pequeñas aplicaciones")

## Applets

- Los applets son programas diseñados para ejecutarse como parte de una página web.
- Java impone restricciones de seguridad para que los applets no puedan "dañar" el ordenador en que se ejecutan

  *Ejemplos*: Acceder a ficheros locales
  Ejecutar otro programa
  Conectarse a otro ordenador desde el nuestro.




- Si los applets se compilasen directamente al código máquina de un ordenador concreto, las personas que accediesen a la página web que contiene el applet desde un ordenador de otro tipo **no podrían ejecutar el applet**.

**Java SDK [Software Development Kit]**


http://java.sun.com

- Compilación de aplicaciones Java: `javac`
- Ejecución de aplicaciones Java: `java`
- Ejecución de applets: `appletviewer`
- Generación de documentación: `javadoc`
- Creación de archivos de distribución JAR [Java ARchives]: `jar`
- Depuración de aplicaciones Java: `jdb`
- Desensamblador para la máquina virtual Java: `javap`
- Generador de stubs en C: `javah`

...

*Versiones*

1995 JDK 1.02
1996 JDK 1.1
1998 JDK 1.2 (Java 2 SDK v1.2)
2000 JDK 1.3 (Java 2 SDK v1.3)
2002 JDK 1.4 (Java 2 Platform, Standard Edition v1.4)
2004 JDK 1.5 (Java 2 Platform, Standard Edition 5.0)

*Ediciones*

**J2SE (Standard Edition):** Aplicaciones y applets
**J2EE (Enterprise Edition):** Servidores de aplicaciones
**J2ME (Micro Edition):** Aplicaciones para dispositivos móviles

**Entornos integrados de desarrollo: IDEs**
[Integrated Development Environments]

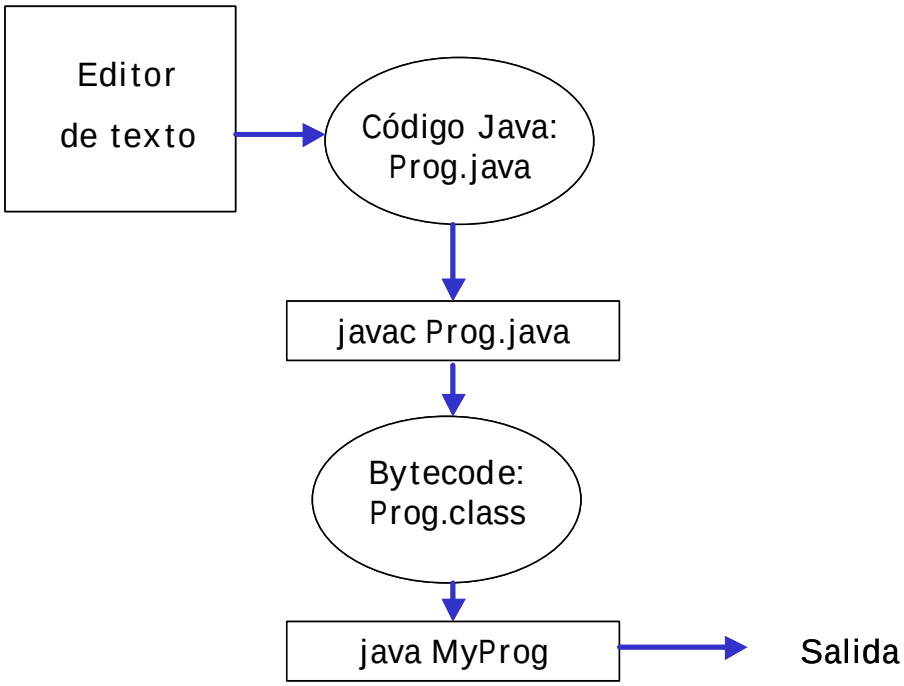
*Gratuitos*

- Eclipse (http://www.eclipse.org)
- NetBeans (http://java.sun.com)

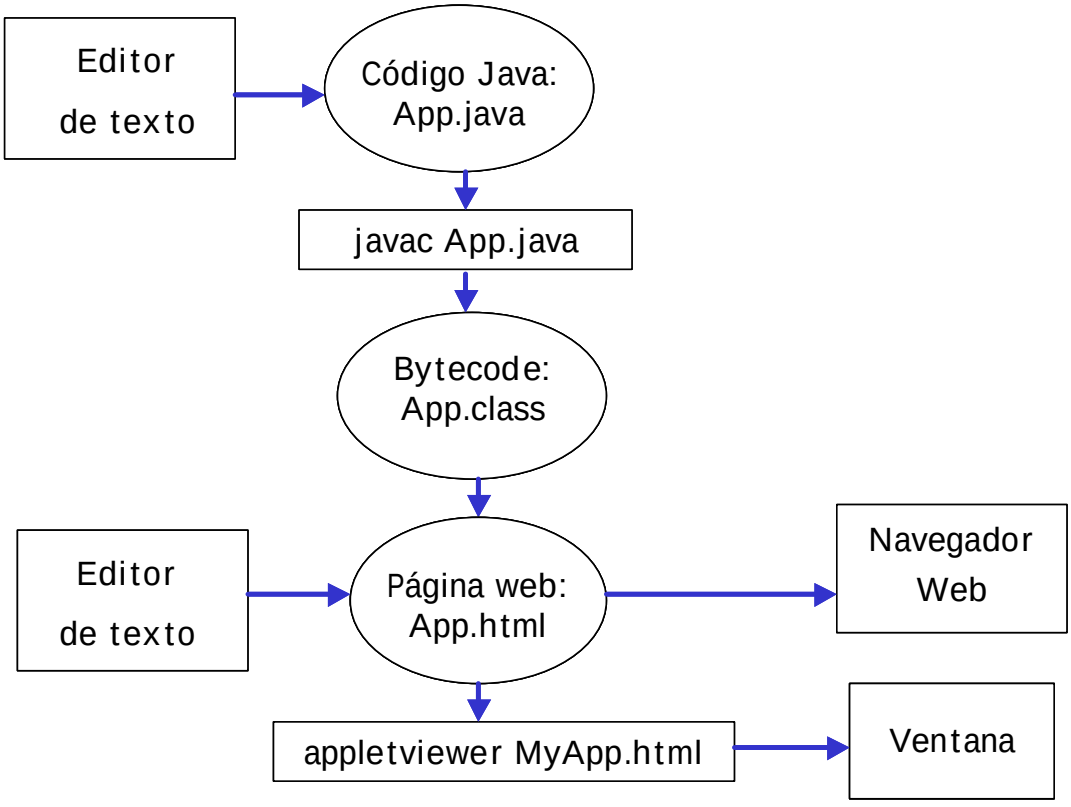
*De pago*

- Borland JBuilder
- IBM Visual Age for Java
- Sybase PowerJ
- Metrowerks CodeWarrior

## Creación y ejecución de aplicaciones Java




## Creación y ejecución de applets



# *Aplicación de ejemplo*

## Código Java: Fichero Programa.java

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

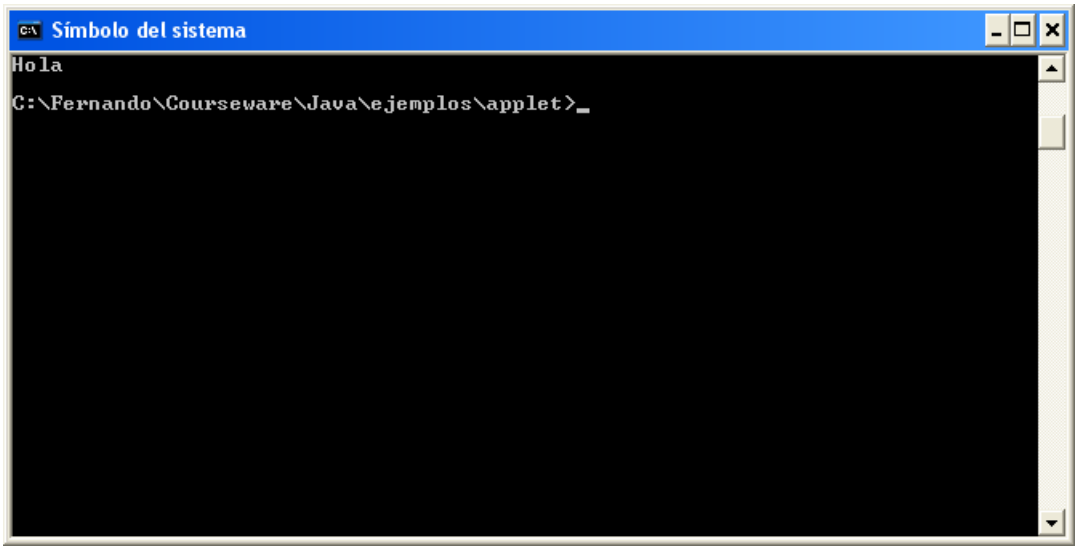
## Compilación

```
javac Programa.java
```

## Ejecución

```
java Programa
```

## Resultado



# *Applet de ejemplo*

## Código Java: Fichero Saludo.java

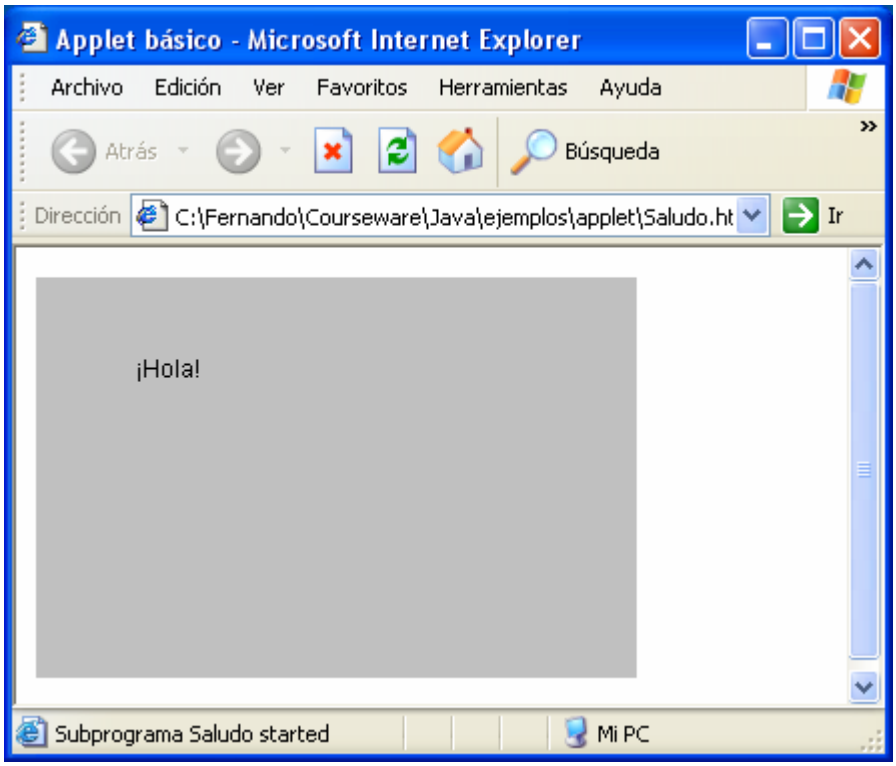
```
import java.awt.*;
import java.applet.Applet;

public class Saludo extends Applet
{
    public void paint(Graphics g) {
        g.drawString("¡Hola!", 50, 50);
    }
}
```

## Página web: Fichero Saludo.html

```
<html>
  <head>
    <title>Applet básico</title>
  </head>
  <body>
    <applet code="Saludo" width=300 height=200>
    </applet>
  </body>
</html>
```

## Resultado



## *Fases en la creación y ejecución de programas en Java*

### Fase I: Editor

- Se crea un programa con la ayuda de un editor
- Se almacena en un fichero con extensión .java

### Fase II: Compilador

- El compilador lee el código Java (fichero .java)
- Si se detectan errores sintácticos, el compilador nos informa de ello.
- Se generan los bytecodes, que se almacenan en ficheros .class

### Fase III: Cargador de clases

El cargador de clases lee los bytecodes (ficheros .class):
Los bytecodes pasan de disco a memoria principal.

### Fase IV: Verificador de bytecodes

El verificador de bytecodes comprueba que los bytecodes son válidos y no violan las restricciones de seguridad de la máquina virtual Java.

### Fase V: Intérprete de bytecodes o compilador JIT

La máquina virtual Java (JVM) lee los bytecodes y los traduce al lenguaje que el ordenador entiende (código máquina).

NOTA: Conforme se ejecuta el programa, se hace uso de la memoria principal para almacenar los datos con los que trabaja la aplicación.

## Java es multiplataforma

Los programas escritos en Java se compilan en un bytecode independiente de la máquina y todos los sistemas operativos principales tienen entornos de ejecución de aplicaciones Java [máquinas virtuales].

NOTA: La idea no es nueva (p.ej. UCSD Pascal)

## Java es seguro

Pueden forzarse restricciones sobre las operaciones permitidas (los applets no acceden directamente al hardware de la máquina).

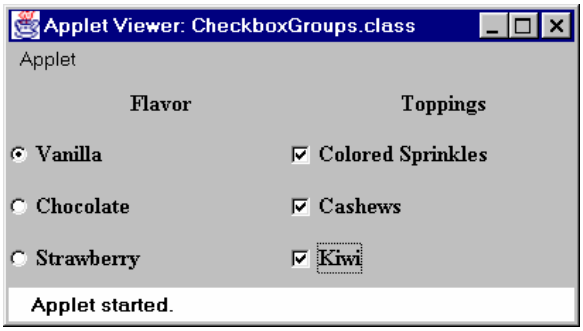
Al cargar un programa en memoria, la máquina virtual Java verifica los bytecodes de la aplicación.

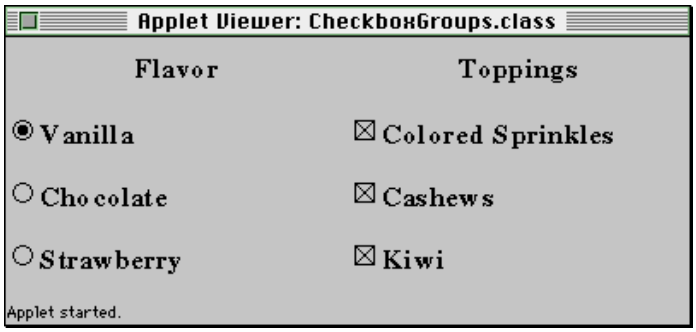
## Java tiene un amplio conjunto de bibliotecas estándar

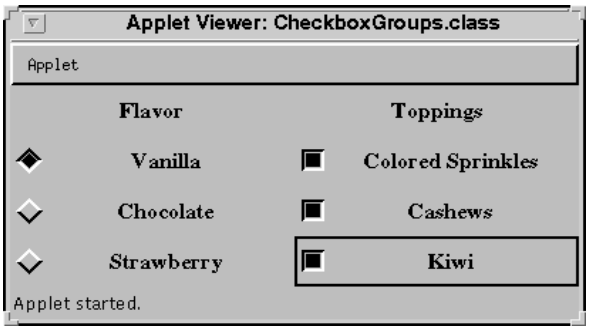
Bibliotecas para trabajar con colecciones y otras estructuras de datos, ficheros, acceso a bases de datos (JDBC), interfaces gráficas de usuario (JFC/Swing), redes de ordenadores (RMI, Jini), aplicaciones distribuidas (EJB), interfaces web (servlets/JSP), hebras, compresión de datos, criptografía...

**Java incluye una biblioteca portable para la creación de interfaces gráficas de usuario (AWT en Java 1.0/1.1 y JFC/Swing en Java 2).**

"Look & feel" en función del sistema operativo:




## Java simplifica algunos aspectos a la hora de programar

- Gestión automática de memoria (recolector de basura).
- Comprobación estricta de tipos
- Sintaxis simplificada con respecto a C++.
  - No se manejan punteros explícitamente (todo son punteros en realidad).
  - No hay que crear makefiles (como en C/C++).
  - No hay que mantener ficheros de cabecera aparte (como en C/C++).
  - No existen macros (#define en C/C++), ya que son propensas a errores.

## *Mitos y realidades de Java*

www.corewebprogramming.com

**Mito:** Java es un lenguaje de programación para la web.
**Realidad:** Java es un lenguaje de programación de propósito general.

*Uso estimado de Java:*
5% applets (clientes web)
45% aplicaciones de escritorio (PCs)
50% aplicaciones en el servidor (servlets/EJB)

**Mito:** "Write once, run anywhere"
**Realidad:** Se puede conseguir, aunque se debe comprobar.

Motivos: Las aplicaciones Java pueden ejecutar código local (nativo), las interfaces gráficas pueden comportarse de forma ligeramente distinta en distintas plataformas...

**Mito:** Java es un lenguaje interpretado.
**Realidad:** Los compiladores JIT compilan el programa al cargarlo.

**Mito:** La seguridad y la independencia de la máquina "son gratis".
**Realidad:** Aplicaciones un 20% más lentas que en C++.

**Mito:** Java acabará con X (donde X puede ser Microsoft, C++...)
**Realidad:** Siempre existen ventajas y desventajas.

Microsoft tiene su propia alternativa: la plataforma .NET

Determinadas aplicaciones es mejor escribirlas en otros lenguajes:
- Utilidades simples y eficientes en ANSI C,
- Sistemas complejos de altas prestaciones en C++,
- Aplicaciones para Windows con Visual Basic .NET o C#...