Java
La Plataforma y el Lenguaje

Java abarca dos aspectos:

- ✓ Una Plataforma.
- ✓ Un Lenguaje de Programación

La Plataforma Java

Una plataforma es un ambiente de software o hardware sobre el que se ejecuta un programa.

La Plataforma Java es sólo una *Plataforma de Software*, que se ejecuta por encima de otras plataformas de software -Sistemas Operativos-.

La plataforma Java tiene dos componentes:

- ✓ La Java Virtual Machine -JVM-.

  La JVM es la base de la Plataforma Java y puede ser incorporada en la mayoría de los plataformas basadas en hardware -Sistemas Operativos-. Contiene el intérprete Java.

- ✓ La Java Application Programming Interface -Java API-

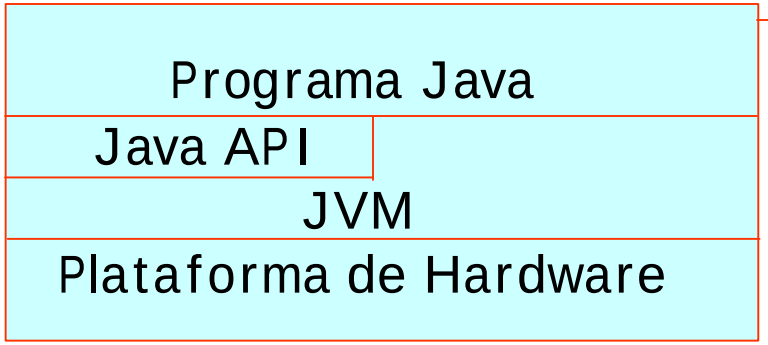
  Es una colección de componentes de software que proveen una amplia gama de funcionalidades, como GUIs, I/O, etc. Está agrupada en paquetes o librerías de componentes relacionadas.

  La Java API se divide en dos grupos:

  - ✓ La API básica
  - ✓ La API extendida

La Plataforma Java
La Java Virtual Machine

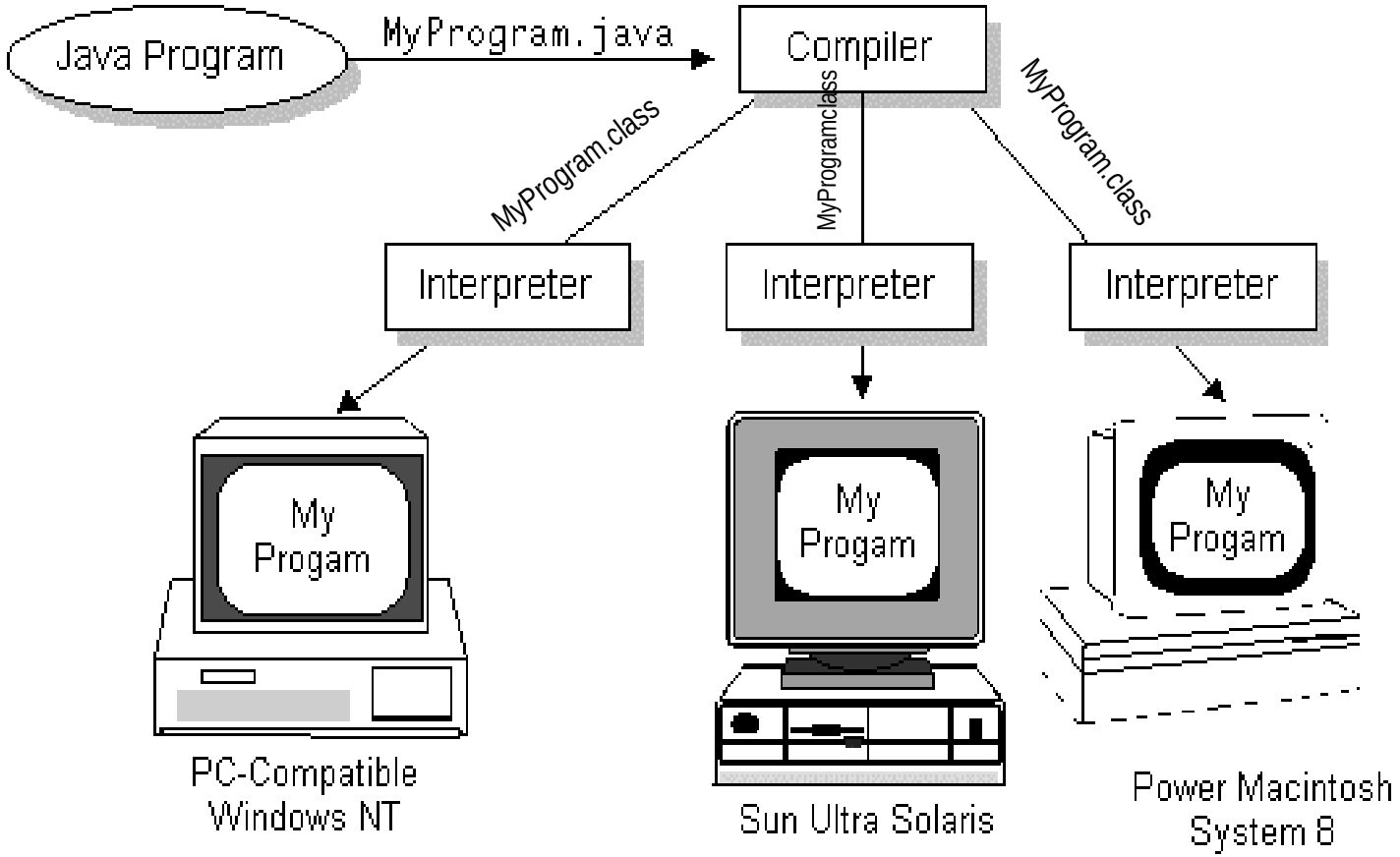
La Plataforma Java -la API Java y la JVM- aislan el programa Java del hardware.



EL código java tiene instrucciones específicas para la JVM

- ✓Es el corazón de la Plataforma Java.
- ✓Oculta a las aplicaciones/applets el Sistema Operativo subyacente.
- ✓Es la clave para la independencia del Sistema Operativo y del hardware subyacente.
- ✓Cada plataforma (SO-Hardware) tiene su propia implementación de la JVM.

La Plataforma Java
La Java Virtual Machine



El programa escrito en JAVA (MyProgram.java), se compila a “bytecodes” (MyProgram.class), y éste puede correr en cualquier lugar donde este presente la plataforma JAVA, sin importar el Sistema Operativo subyacente. La *Java Virtual Machine* asegura esta portabilidad.

La Plataforma Java
La Java API Básica

La librería básica de clases Java, está formada por los siguientes paquetes:

- ✓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 reproduccián 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).

Las nuevas versiones de la Plataforma se obtienen, migrando librerías desde la API extendida y mejorando las librerías existentes.

El Lenguaje Java

Java posibilita el desarrollo de aplicaciones *seguras*, de *alta performance*, *robustas* sobre *múltiples plataformas* en redes *heterogéneas y distribuídas*.

Las principales características del lenguaje Java son:

- ✓ Simple y familiar
- ✓ Orientado a objetos
- ✓ Distribuído
- ✓ Robusto
- ✓ Seguro
- ✓ Arquitectura Neutral
- ✓ Multithreaded
- ✓ Alta performnace
- ✓ Interpretado
- ✓ Dinámico

El Lenguaje Java

Simple y Familiar

El objetivo de los diseñadores de Java fue crear un lenguaje de programación que fuera fácil de aprender. Java adoptó una sintaxis similar a la de C/C++ teniendo en cuenta su popularidad, y eliminó aquellas características que son fuente de confusión.

Principales características de Java

- ✓ Tipos de datos primitivos (float, int, long, short, double, byte, char y boolean)
- ✓ Operadores aritméticos y relacionales
- ✓ Arreglos
- ✓ Strings
- ✓ Sentencias Break y Continue con etiquetas
- ✓ Manejo de Memoria y Garbage Collection
- ✓ Thread sincronizados

Características de C/C++ eliminadas en Java

- ✓ No más preprocesador
- ✓ No más estructuras ni uniones
- ✓ No más tipos enumerativos
- ✓ No más funciones
- ✓ No hay herencia múltiple de clases
- ✓ No más GOTO
- ✓ No más operadores sobrecargados
- ✓ No más punteros

El Lenguaje Java
Orientado a Objetos

- ✓ Java posee todas las características de un lenguaje orientado a objetos
  - ✓ Polimorfismo.
  - ✓ Encapsulamiento.
  - ✓ Binding Dinámico.
  - ✓ Herencia.

- ✓ Java implementa herencia simple de clases. Una clase puede ser subclase de una única clase.Todas las clases heredan de Object.

- ✓ Java enriquece el modelo de herencia simple de clases, implementando *interfaces.* Una interfaz es una especificación de constantes y métodos sin implementación. Una clase puede implementar una o más interfaces, logrando una *similitud* con herencia múltiple.

El Lenguaje Java
Distribuído

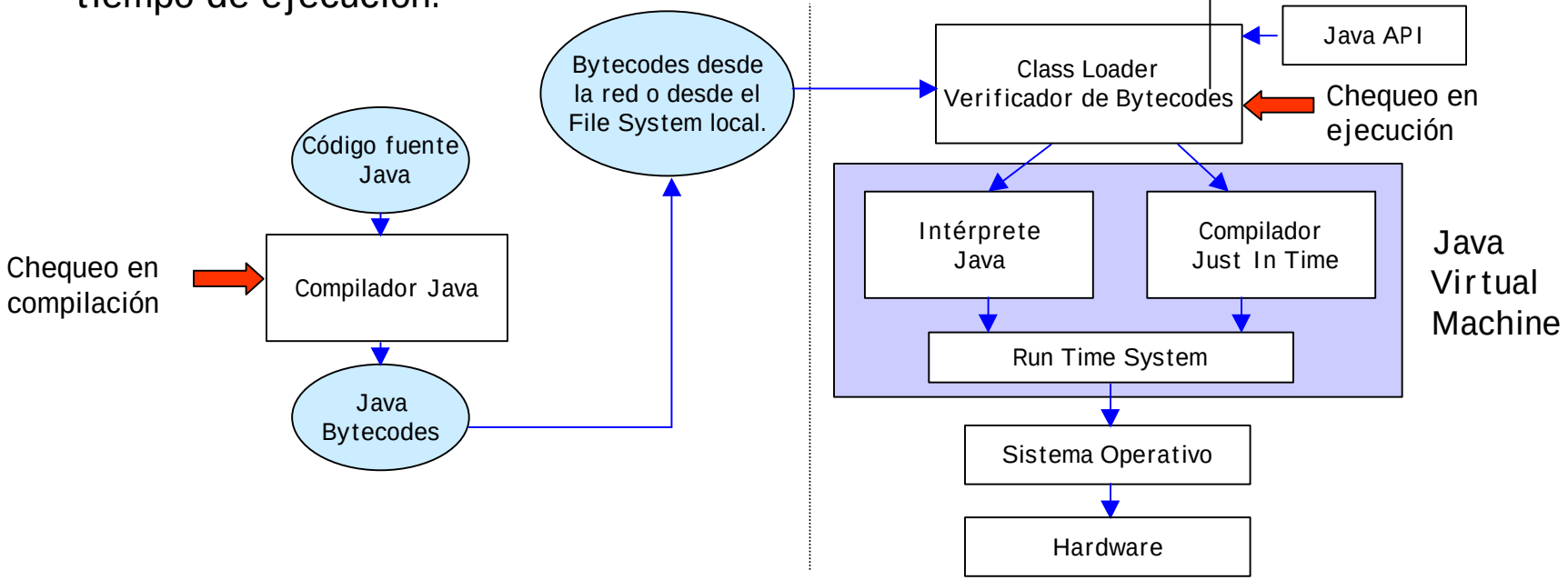
- ✓ Con Java es posible usar protocolos como HTTP y FTP para copiar archivos remotos de manera tan simple como si estuviesen en el File System Local.
- ✓ El comportamiento distríbuido de Java posibilita la colaboración y la distribución de carga de trabajo del sistema. Ejemplo: Applets.
- ✓ RMI (Java Remote Method Invocation) provee una forma simple y directa de manejar objetos Java distribuídos.

El Lenguaje Java

Robusto

Verifica que en el codigo no venga un código que extienda a una clase final, por ejemplo.

- ✓ Java fue diseñado para la creación de software altamente confiable. Provee un chequeo en tiempo de compilación, seguido por un segundo nivel de chequeo en tiempo de ejecución.



- ✓ Java provee un modelo de manejo de memoria extremadamente simple: no hay punteros definidos explícitamente y provee una recolección de basura automática -Garbage Collection-. Así se eliminan fuentes de errores, roturas de sistemas y baja performance.

El Lenguaje Java
Seguro

- **Mapa de memoria y Alocación de memoria:**
  - En Java la reserva de memoria se realiza en ejecución y depende de las características del software y hardware de la estación de trabajo en donde se ejecuta el programa.
  - En Java no existe el concepto de punteros a memoria al estilo C y C++. El código Java compilado hace referencia a la memoria vía “manejadores simbólicos”, que se resuelven a memoria real en ejecución.
  - En Java el modelo de alocación de memoria es transparente al programador, ya que es controlado integramente por JVM.
  - En Java los programadores no pueden falsificar ni construir punteros a memoria, contribuyendo a la seguridad de las aplicaciones.
- **Chequeos de seguridad en el Class Loader:**
  - Las clases de la API Java no pueden ser sobreescritas por clases importadas desde la red. Las clases importadas desde la red, se ubican en un espacio de nombres privados. En el momento en que la clase referencia a otra clase, ésta se busca primero en el File System Local y, luego en el espacio de nombres privado.
- **Verificación del ByteCode:**
  - El intérprete Java chequea los archivos .class que vienen de la red, evaluando:
    - que el código no falsifique punteros.
    - que el código no viole restricciones de acceso.
    - que el código no viole el acceso a los objetos usando casting.

El Lenguaje Java

Arquitectura Neutral

“Write Once, Run Anywhere”

✓Java fue diseñado para soportar aplicaciones que se ejecutan en ambientes de redes heterogéneos, independientemente de la plataforma de hardware y de software (Sistema Operativo).

✓Por ello, las aplicaciones escritas en Java se ejecutan en una amplia variedad de arquitecturas de hardware y, sobre múltiples sistemas operativos.

✓Es el formato de “bytecodes” de Java el que permite que las aplicaciones se ejecuten en una amplia diversidad de ambientes de ejecución. El formato de “bytecodes” es un formato intermedio de arquitectura neutral que permite transportar eficientemente código entre múltiples plataformas de hardware y software.

✓El mismo código en formato “bytecode” se ejecuta sobre cualquier plataforma de hardware y software que disponga de la JVM (donde reside el intérprete Java).

La arquitectura neutral dada por los “bytecodes” es el paso más importante hacia la **portabilidad** de los programas

El Lenguaje Java

Multithread

Un **Thread** es un flujo de control secuencial dentro de un programa. Java provee múltiples threads en un programa, ejecutándose concurrentemente y llevando a cabo tareas distintas.

✓ La API Java contiene la clase Thread que soporta una conjunto de métodos para tratar threads: arrancar un thread (start), ejecutar un thread (run), interrumpir la ejecución de un thread (stop) y chequear el estado de un thread.

✓ El soporte de Threads en Java incluye primitivas de sincronización. Si múltiples threads ejecutándose concurrentemente comparten recursos, necesitan mecanismos de sincronización para acceder al recurso. Esto se logra, declarando a los métodos *synchronized.*

Multithread permite mejorar la interactividad y la performance del sistema.

El Lenguaje Java

Alta Performance

✓El código Java es chequedo en compilación y en ejecución (por el Class Loader y ByteCode Verifier) por lo tanto el intérprete puede correr a toda velocidad sin necesidad hacer chequeos en ejecución.

✓El garbage collection automático corre como un thread de baja prioridad (aprovechando los tiempos muertos del usuario), mejorando la disponibilidad de memoria.

El Lenguaje Java

Interpretado y Dinámico

- ✓ El compilador Java genera “bytecodes” para la JVM. El intérprete, incorporado en la JVM es el que permite ejecutar el programa.
- ✓ Los “bytecode” de Java pueden ejecutarse en cualquier plataforma que tenga la JVM implementada.
- ✓ En Java no existe el proceso de linkedición. Este se reemplaza por la carga de nuevas clases desde el Class Loader (contenido en la JVM).
- ✓ La naturaleza interpretada de Java acelera el ciclo de desarrollo de software. Permitiendo rápida prototipación facilitando el testeo.
- ✓ Java es dinámicamente extensible ya que las clases se linkean a medida que se necesitan y pueden ser cargadas dinámicamente a través de la red.