

# Laboratorio de Software

# Cursada 2025

**Profesores:** Claudia Queiruga y Pablo Iuliano

**JTP:** Diego Bellante e Isabel Miyuki Kimura

**Ayudantes:** Andrés Vazzano, Francisco Blanco, Sebastián Villena, Sebastián Perri



# ¿Qué vamos a aprender? ¿Cómo nos vamos a organizar? ¿Cómo se promociona?

## Objetivos de aprendizaje

- Abordar un **proyecto de software** que signifique la **aplicación concreta de los conocimientos adquiridos hasta el momento en la carrera**.
- Desarrollar un **aplicación móvil nativa en Android** sobre una problemática concreta enmarcada en una situación real con adoptantes/usuarios reales (trabajo final integrador).

## Metodología de trabajo

A lo largo de la cursada se trabajarán:

- **contenidos teóricos** en clases **presenciales** que se articulan con los prácticos.
- **trabajos prácticos** y **talleres** en los que se pondrán en común los temas trabajados, con entregas que se defenderán en **coloquios**.
- el **desarrollo de un proyecto final integrador, en equipo**, con 3 entregables en fechas pautadas, en clases prácticas (presencial).

Cada **equipo** de estudiantes tendrá un docente asignado que lo tutorará.



# ¿Qué vamos a aprender? ¿Cómo nos vamos a organizar? ¿Cómo se promociona?

## Metodología de evaluación

- Promoción directa.
- Durante la cursada se realizarán 5 instancias de evaluación parcial: 2 individuales y 3 grupales (3 entregables del proyecto integrador).
- Se promociona (sin final!!) con al menos nota 6 de promedio entre las 5 evaluaciones parciales y al menos nota 4 en cada evaluación.
- Los estudiantes que no alcancen la nota de promoción pero que hayan aprobado al menos con 4 puntos cada una de las evaluaciones parciales, aprueban la cursada y deberán rendir un examen final en una mesa examinadora.



# Herramientas y tecnologías usaremos



# La Plataforma Java






# JAVA

## La Plataforma y el Lenguaje

**Java abarca dos aspectos:**

**Una Plataforma de Software**

**Un Lenguaje de Programación**

- JAVA fue desarrollado por SUN Microsystems a principios de los años 90. James Gosling, el diseñador original de JAVA, implementó el compilador JAVA y la JVM (Java Virtual Machine).
- A partir de 2010 JAVA fue adquirido por Oracle.
- A partir de 2017 modificó el modelo de lanzamiento de nuevas versiones:
  -  Release cada 6 meses (feature releases).
  -  Parches de seguridad cada 3 meses.
  -  LTS cada 3 años, con soporte extendido por 8 años o más.
- La gobernanza de la evolución de la plataforma JAVA está controlado por el [Java Community Process \(JCP\)](#).
- Actualmente JAVA está entre los 10 lenguajes de programación más usados:
  - IEEE Spectrum: <https://spectrum.ieee.org/top-programming-languages/>
  - Índice TIOBE: <https://www.tiobe.com/tiobe-index/>
  - Encuesta de Stackoverflow: <https://survey.stackoverflow.co/2025/technology>

# La Plataforma JAVA



La plataforma JAVA “**está disponible en todos lados**”. Los 3 sabores de JAVA:

## Java Standard Edition (JSE)

Está diseñado para el desarrollo de aplicaciones de escritorio. Se ejecuta sobre todos los SO. Provee la infraestructura de compilación y de ejecución a través de la máquina virtual y las API básicas. La **API de JSE ofrece la funcionalidad principal del lenguaje JAVA**, define todos los tipos básicos y objetos JAVA hasta clases de alto nivel para ser usadas en conectividad, seguridad, acceso a BD, desarrollo de GUI, parsing XML, etc.

## Java Enterprise Edition (JEE)

Java del lado del servidor. Actualmente se llama **Jakarta EE** y su desarrollo es liderado por Eclipse Foundation.

## Java Micro Edition (JME)

Es una versión simplificada de JSE para dispositivos móviles. La falta de acceso al hardware del dispositivo y la ejecución en un entorno controlado (sandbox) da como resultado que las aplicaciones **JME no aprovechan las ventajas propias del mundo móvil. NO fue adoptada como plataforma de desarrollo de aplicaciones móviles nativas. Android es la plataforma de desarrollo y ejecución de aplicaciones móviles nativas en JAVA y KOTLIN.**

# Java es Estándar

- Las tecnologías Java tienen una especificación desarrollada por el Java Community Process (JCP): <http://jcp.org/en/home/index>
- **Java Community Process** es la comunidad de desarrollo de especificaciones de tecnologías JAVA. **Gobierna la evolución de JAVA**, aprueba y estandariza.
- Contar con una especificación transforma a las **tecnologías JAVA en estándares**. Cada fabricante de software desarrolla su implementación respetando la especificación. De esta manera se garantiza compatibilidad y portabilidad.
- Las especificaciones para JSE, junto con las APIs asociadas, son desarrolladas por el JCP. La evolución de JEE es controlada por Eclipse Foundation.
- Los **JEP** (JDK Enhancement Proposal) son **documentos técnicos** gestionados dentro de **OpenJDK** que proponen **mejoras en Java**. Muchas de estas propuestas se formalizan vía JCP. Ej. la palabra clave `var` para inferencia de tipos de variables locales.
- **OpenJDK** es la implementación de referencia de Java SE de **código fuente abierto** (<http://openjdk.java.net/>) con licencia GPL.
- Una especificación comienza como un Java Specification Request (JSR), que pasa por varios estados en la JCP antes de convertirse en la especificación definitiva. Cada JSR tiene asignado un número:
  - JSR 384: JSE 11 (LTS, septiembre 2018)**
  - JSR 386, 388, 389, 390, 391: JSE 12, 13, 14, 15 y 14 (no LTS)**
  - JSR 392: jse 17 (LTS, septiembre 2021)**
  - JSR 393: jse 18 (no LTS, marzo 2022)**
  - JSR 399: jse 24 (no LTS, marzo 2025)**





# Plataformas de Ejecución

Una **plataforma**, en términos generales, es un sistema formado por un **hardware (arquitectura)** y un **sistema operativo junto a ciertas librerías de ejecución**, que permiten la ejecución de aplicaciones, programas o procesos. Es la combinación de hardware y software que sirve de soporte para *alojar y ejecutar* aplicaciones o servicios. Es el **escenario sobre el que se ejecutan los programas**.

Una **plataforma** cumple con un conjunto de estándares que permite a los desarrolladores de software desarrollar aplicaciones para dicha plataforma.

**JAVA es una plataforma sólo de software que se ejecuta por encima de otras plataformas que combinan hardware y software.**

Los **programas escritos en JAVA** son independientes del sistema operativo y del hardware donde se ejecutan. Se **ejecutan sobre la plataforma JAVA**.

La **plataforma de ejecución de JAVA se llama JRE** (Java Runtime Environment) y provee todos los componentes necesarios para ejecutar programas JAVA. Su componente más importante es la **JVM (Java Virtual Machine)**.

A su vez **JSE (Java Standard Edition)** es una plataforma que ofrece todas las herramientas y librerías necesarias para desarrollar programas JAVA y ejecutarlos en la JRE.



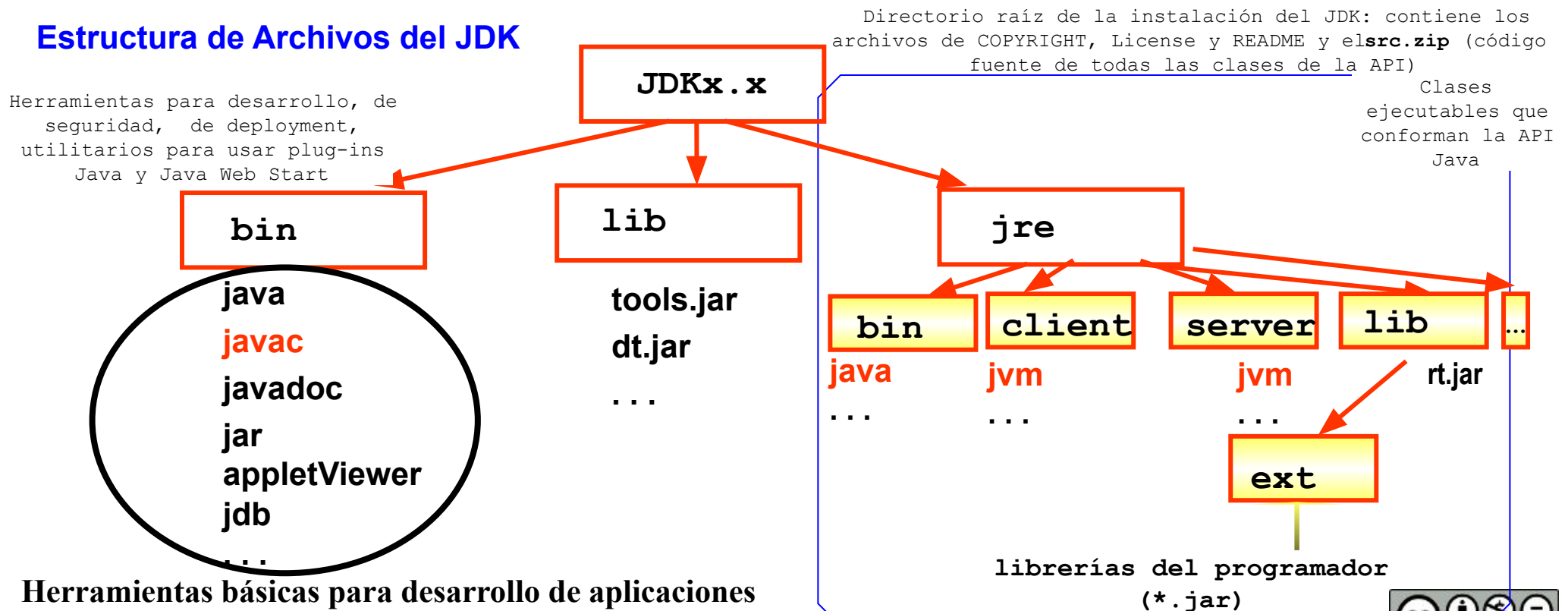
# La Plataforma JSE

El JSE está compuesto por una plataforma de desarrollo llamada **JDK** y una de ejecución llamada **JRE**.

**Java Runtime Environment (JRE)** está compuesto por librerías de componentes de software (JAVA API), la Máquina Virtual (HotSpot VM) y otras componentes necesarias para ejecutar y desplegar aplicaciones de escritorio escritas en JAVA.

**Java SE Development Kit (JDK)** es un superconjunto de JRE, contiene todo lo que está en JRE más herramientas de desarrollo como un compilador, debugger, compactador, documentador, necesarios para desarrollar aplicaciones de escritorio. Herramientas de seguridad, de deployment.

## Estructura de Archivos del JDK



# Componentes de la Plataforma JSE

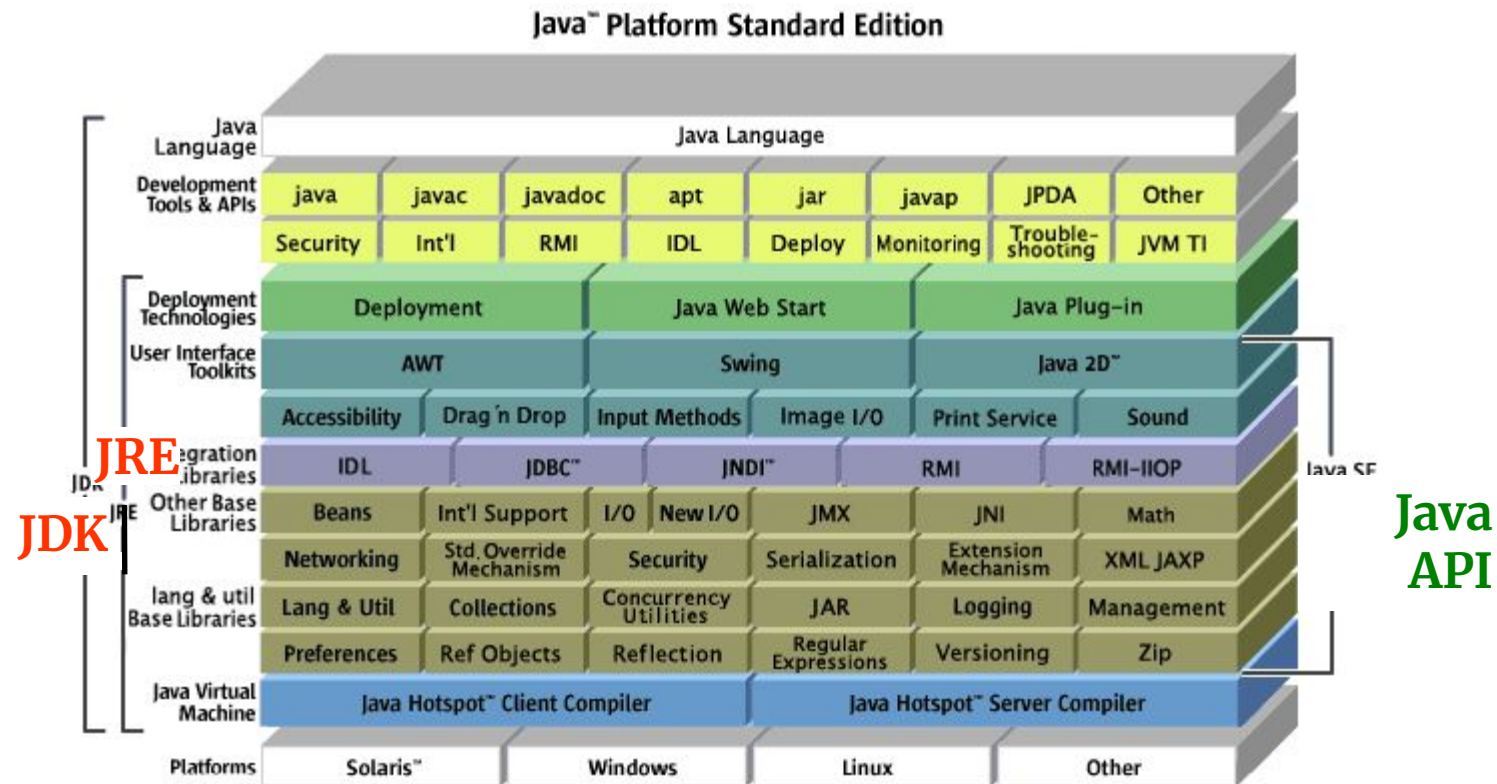


Gráfico extraído de <http://www.oracle.com/technetwork/java/javase/tech/index.html>

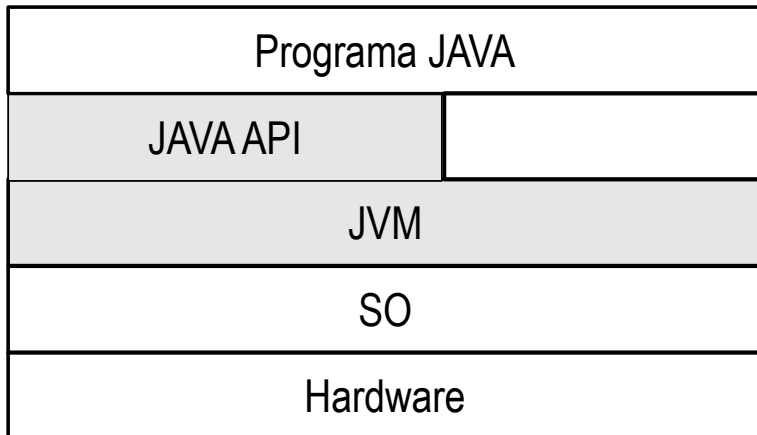
La Máquina Virtual Java **HotSpot es una componente fundamental** de la plataforma Java SE. Implementa la especificación de la JVM, se distribuye junto al JRE.

Usa **compilación dinámica adaptativa** es decir el código no se compila todo de una vez sino durante la ejecución y a la vez “aprende” qué partes del código son las más usadas y lo optimiza dinámicamente.

La plataforma JSE contiene 2 implementaciones de la VM HotSpot: **HotSpot Client VM** y **HotSpot Server VM**

# Java Virtual Machine

La JVM es una máquina de software que emula una máquina real



El código Java compilado tiene instrucciones específicas para la JVM

- La JVM aísla al programa Java del SO y del hardware sobre el que se ejecuta.
- La JVM es responsable de la “independencia” de la plataforma (SO + Hardware), de los bytecodes y de la seguridad.
- Cada S.O. tiene su propia implementación de la JVM.
- La especificación de la JVM provee definiciones para: el conjunto de instrucciones (es un equivalente al conjunto de instrucciones de la CPU), conjunto de registros de máquina, el formato de archivos .class, la pila de ejecución, una heap con garbage-collection, un área de memoria donde instalarse.

# La Plataforma Java

## Herramientas

La plataforma JSE es usada en una amplia variedad de herramientas, entre ellas: entornos de desarrollo integrados o IDEs (Integrated Development Environments), herramientas de testeo, de monitoreo de performance, servidores de aplicaciones, etc.

| IDE                           | Software Libre y Código Fuente Abierto | Propietario |
|-------------------------------|--|-------------|
| Eclipse                       | ✓                                      |             |
| NetBeans                      | ✓                                      |             |
| IntelliJ IDEA                 | ✓                                      | ✓           |
| BlueJ                         | ✓                                      |             |
| Android Studio<br>(no es JSE) | ✓                                      |             |
| Oracle<br>JDeveloper          |  | ✓           |



ANDROID



# La API JAVA

La interface de programación JAVA o API (Application Programming Interface) es un conjunto de librerías de clases e interfaces compiladas, listas para usar, organizadas en paquetes, que forman parte de la distribución del JSE.

## Algunos paquetes de la 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, excepciones, etc.

**java.net:** contiene clases como URL, TCP, UDP, IP, etc. que permiten implementar aplicaciones distribuidas. 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.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).

**ETC.....**



# Android, JAVA y KOTLIN

**Android** es un **sistema operativo de código fuente abierto** (controlado por Google) para dispositivos móviles **basado en el núcleo de Linux**. Actualmente está presente en una amplia gama de dispositivos: Android WEAR, Android AUTO, Android TV, etc.

Windows Mobile y iPhone de Apple ofrecen un rico entorno de desarrollo para aplicaciones móviles. Sin embargo, a diferencia de Android, están basados en sistemas operativos propietarios que frecuentemente dan prioridad a las aplicaciones nativas sobre las creadas por terceras partes y restringe la comunicación entre las aplicaciones y los datos nativos de teléfono.

**Android** ofrece nuevas **posibilidades para aplicaciones móviles** al estar basado en un entorno de desarrollo abierto como Linux. El acceso al hardware está disponible para todas las aplicaciones a través de una serie de APIs y soporta interacción entre aplicaciones.

Las **aplicaciones nativas en Android** se pueden escribir en **JAVA** o **Kotlin**, sin embargo Google anunció en 2019 que **Kotlin es el lenguaje preferido para desarrollar aplicaciones nativas en Android**. **Kotlin es totalmente interoperable con Java**.

