Programación Orientada a Objetos

Tema 1: Introducción

Eduardo Mosqueira Rey

LIDIA
Laboratorio de Investigación y desarrollo en Inteligencia Artificial

Departamento de Computación
Universidade da Coruña, España

Objetivos

- **Contextualizar el paradigma de la orientación a objetos dentro de los distintos paradigmas de programación existentes.**
- **Analizar las diferencias metodológicas entre la programación por descomposición funcional y la programación orientada a objetos.**
- **Introducir al alumno en el contexto de los lenguajes orientados a objetos.**
- **Conocer la historia y los aspectos básicos tanto del lenguaje como de la plataforma Java.**

Índice

Índice

Paradigmas Programación

Especifican la programación como un conjunto de secuencias computacionales que se ejecutan paso a paso

No existe una descripción paso a paso de cómo llegar a la solución, sino que el programa se construye estableciendo hechos, reglas, restricciones, ecuaciones, etc. Que debe cumplir el conjunto de valores que constituyen la solución.

Paradigmas de programación

Colección de patrones conceptuales que juntos modelan el proceso de diseño y, finalmente, determinan la estructura del programa.
Paradeigma = Modelo

Se provee al sistema de soluciones a problemas similares y se deja que generalice una solución a partir de estos ejemplos.

Paradigma operacional

- Paradigma imperativo
- Paradigma de orien-tación a objetos

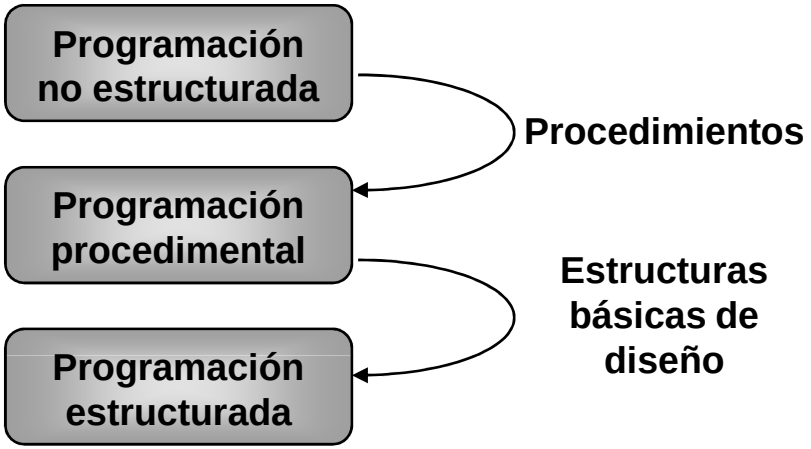
Paradigma declarativo

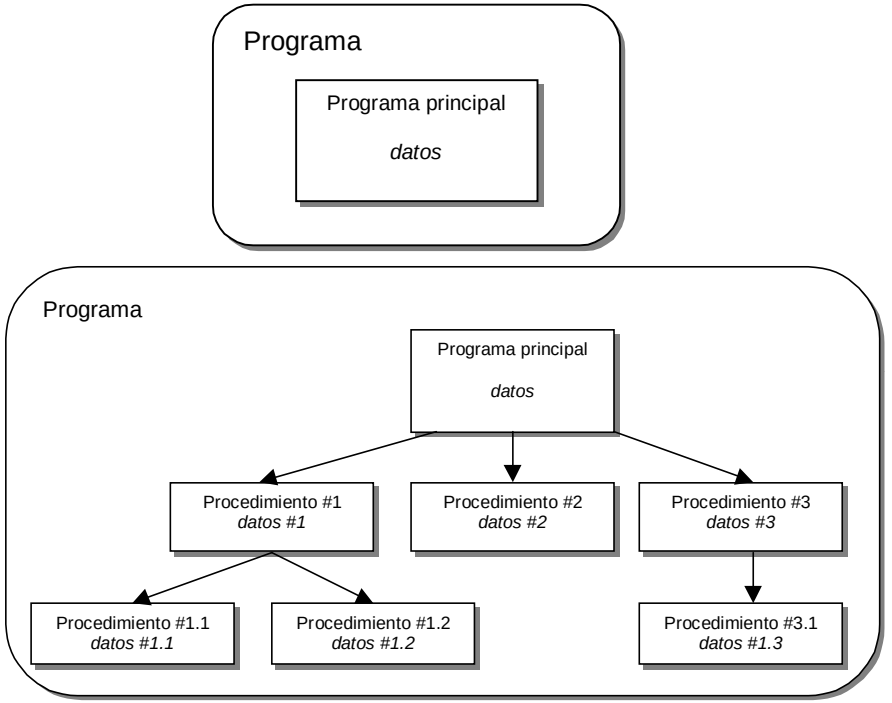
- Paradigma funcional — Haskell, Erlang
- Paradigma lógico — Prolog
- Paradigma transformacional — Sistemas basados en reglas
- Paradigma relacional — SQL

Paradigma demostrativo

- Paradigma de inducción
- Paradigma de redes de neuronas
- Paradigma genético

Evolución Lenguajes Imperativos





Evolución Lenguajes Imperativos

Programación no estructurada

Procedimientos

Programación procedimental

Estructuras básicas de diseño

Programación estructurada

Encapsulación en módulos

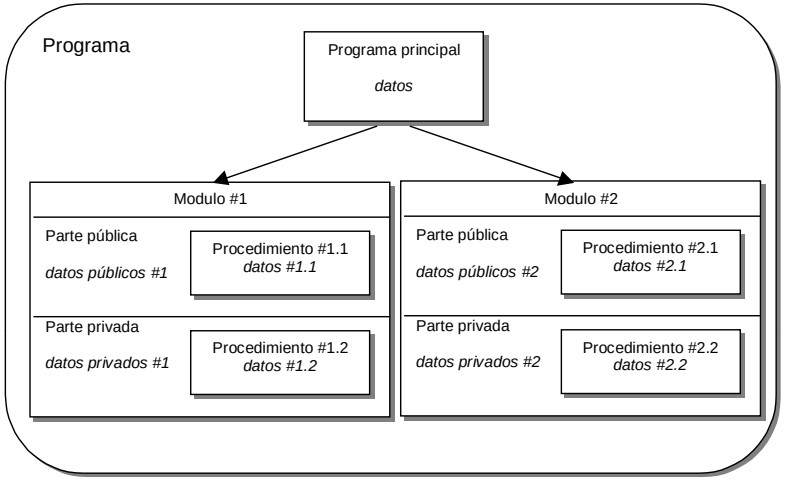
Programación modular

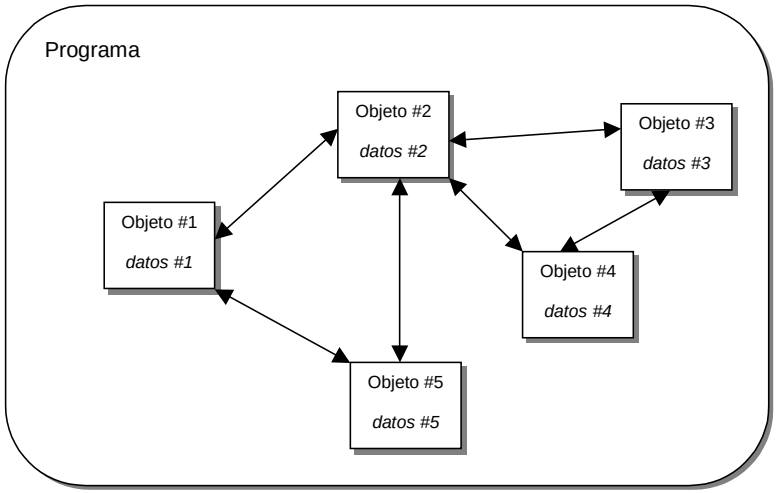
Instanciación

Tipos abstractos de datos

Herencia, polimorfismo, etc.

Programación orientada objetos





Índice

3. Programación orientada a objetos

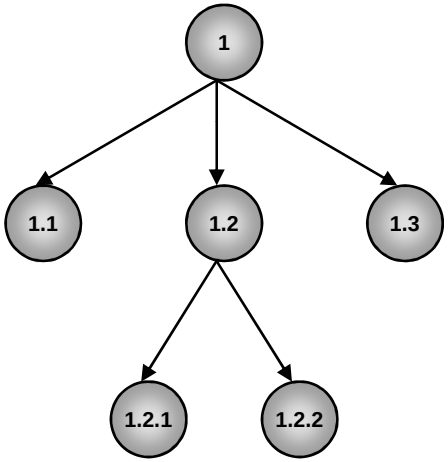
- Introducción
- Top-down vs. Bottom-up
- Ventajas de la POO
- Inconvenientes de la POO

Prog. Orientada a Objetos
Introducción

- **Características**
  - **Cambio de notación:**
    - **tipos → clases**
    - **variables → objetos.**
  - **Nuevas características OO no presentes en los TADs**
    - **Herencia:**
      - **Las clases pueden establecer relaciones de generalización-especialización de forma que, implícitamente, las clases especializadas hereden propiedades de las clases genéricas**
    - **Polimorfismo:**
      - **Capacidad de una subclase de actuar como si fuera su superclase**
  - **La nueva filosofía de programación se orienta a la estructura de los datos en vez de a la estructura de las funciones**

Prog. Orientada a Objetos
Top-down vs. Bottom-up

- **Diseño top-down o descomposición funcional**
  - **Características**
    - **Consiste en ir descomponiendo el programa en piezas más pequeñas y manejables (subrutinas, funciones o procedimientos).**



  - **Problemas**
    - **No favorece la reutilización porque las funciones de más bajo nivel desarrolladas son muy dependientes del problema que pretenden resolver y de los datos globales existentes en el programa en el que se incluyen.**
    - **Se ve muy afectado por cambios en los requisitos funcionales.**

Prog. Orientada a Objetos
Top-down vs. Bottom-up

- **Diseño Bottom-up**
  - **Filosofía de la orientación a objetos**
    - **Se trata de identificar cómo es la estructura de datos del programa y qué interacciones aparecen entre los distintos datos.**
    - **A partir de los datos y sus interacciones se desarrollan las funciones que produzcan las salidas adecuadas**
    - **El programa se compone de una serie de objetos con un estado interno propio y que interactúan intercambiando mensajes**
  - **Ejemplo**
    - **Para realizar un juego de cartas empezamos realizando los objetos carta, mazo, tapete, jugador, etc.**
    - **Los objetos se prueban de forma independiente y posteriormente se integran para formar el juego (bottom-up)**
    - **Los objetos son fácilmente reutilizables para cualquier otro juego de cartas**

Prog. Orientada a Objetos
Ventajas

- **Reutilización**
  - **Los objetos bien diseñados pueden utilizarse como base de otros sistemas, que se constituyen como una combinación de los objetos existentes.**
- **Modularidad**
  - **Los objetos son autocontenidos y tienen definido de forma clara sus interfaces con otros objetos.**
- **Comprensión**
  - **Al estar los datos y procedimientos que conforman los objetos encapsulados en un mismo compartimento, los objetos pueden ser desarrollados y probados en forma independiente.**

Prog. Orientada a Objetos
Ventajas

- **Extensibilidad**
  - **La herencia permite que sea posible definir y utilizar de forma clara módulos funcionalmente incompletos y, luego, permiten su extensión sin trastornar la operación de los módulos cliente.**
- **Escalabilidad**
  - **Los diseños orientados a objetos, al ser fácilmente extensibles, permiten que el esfuerzo no aumente exponencialmente con el tamaño y la complejidad del proyecto, tal y como sucede con los sistemas convencionales**
- **Naturalidad**
  - **El análisis y el diseño que dividen un dominio en objetos, está más acorde con la realidad que una descomposición funcional por refinamiento progresivo.**

Prog. Orientada a Objetos
Inconvenientes

- **Curvas de aprendizaje largas:**
  - **Aprender un lenguaje orientado a objetos y dominar sus técnicas fundamentales es más complejo que aprender un lenguaje imperativo tradicional.**
- **Cambio de enfoque:**
  - **Los desarrolladores generalmente aprenden a programar con lenguajes imperativos a través de una filosofía top-down y deben cambiar a una filosofía bottom-up**
- **Reutilizaciones ineficientes:**
  - **Los objetos en sí mismos no son reutilizables por ser objetos, hay que programar pensando en la reutilización**
  - **Si no está claro que el objeto se vaya a reutilizar no siempre se invierte el esfuerzo extra en hacerlo reutilizable**

Prog. Orientada a Objetos
Inconvenientes

- **Rendimiento:**
  - **Cuanto más nos alejemos del código máquina que finalmente se ejecuta, más recursos consumirán los programas y más dependeremos de los procesos de optimización de los compiladores**
  - **Este problema se mitiga con el constante avance en las prestaciones del hardware**

- **Orientación a objetos forzada:**
  - **“Si la única herramienta que tienes es un martillo, entonces tiendes a ver todos los problemas como si fueran clavos”.**
  - **Forzar la utilización de objetos en determinados entornos puede producir diseño no adecuados**
    - **Java presenta tipos básicos (int, float) que no son objetos**
    - **La clase Math definida en el API de Java se basa en métodos de clase (static en Java), algo muy poco OO.**

Índice

Lenguaje Java
Historia

- **Origen**
  - **Lenguaje Oak, desarrollado en Sun por James Gosling y Peter Naughton para controlar pequeños dispositivos electrónicos**
  - **Aspectos buscados: independencia del hardware y lenguaje OO puro y robusto basado en C++**
  - **Puro:**
    - **No se admiten construcciones (métodos, variables, etc.) que caigan fuera del ámbito de una clase**
  - **Robusto**
    - **Se eliminaron aquellos aspectos que hacían a C++ propenso a errores (manejo de punteros, comprobación no estricta de tipos, manejo de memoria, etc.).**
  - **Cancelado en la primavera de 1994 ante la falta de beneficios (se consideró que era muy ambicioso para su época)**

Lenguaje Java
Historia

- **Aparición de Internet (1995)**
  - **Cambio de nombre (Oak ⇒ Java) y de diseño**
    - **Oak se llamó así por un roble cercano al laboratorio, como estaba licenciado se cambió por Java un día que bajaron a la cafetería ⇒ símbolo de Java: una taza de café, componentes Java: JavaBeans (granos de café)**
  - **1995: Navegador HotJava**
    - **Capaz de descargar pequeños programas de Internet (applets) y ejecutarlos de forma segura en nuestro ordenador.**
  - **1998: Plataforma Java 2 (v1.2)**
    - **Corrige los principales fallos del programa**
    - **Se le da la estabilidad y la continuidad requerida a las bibliotecas**

Lenguaje Java
Plataforma Java

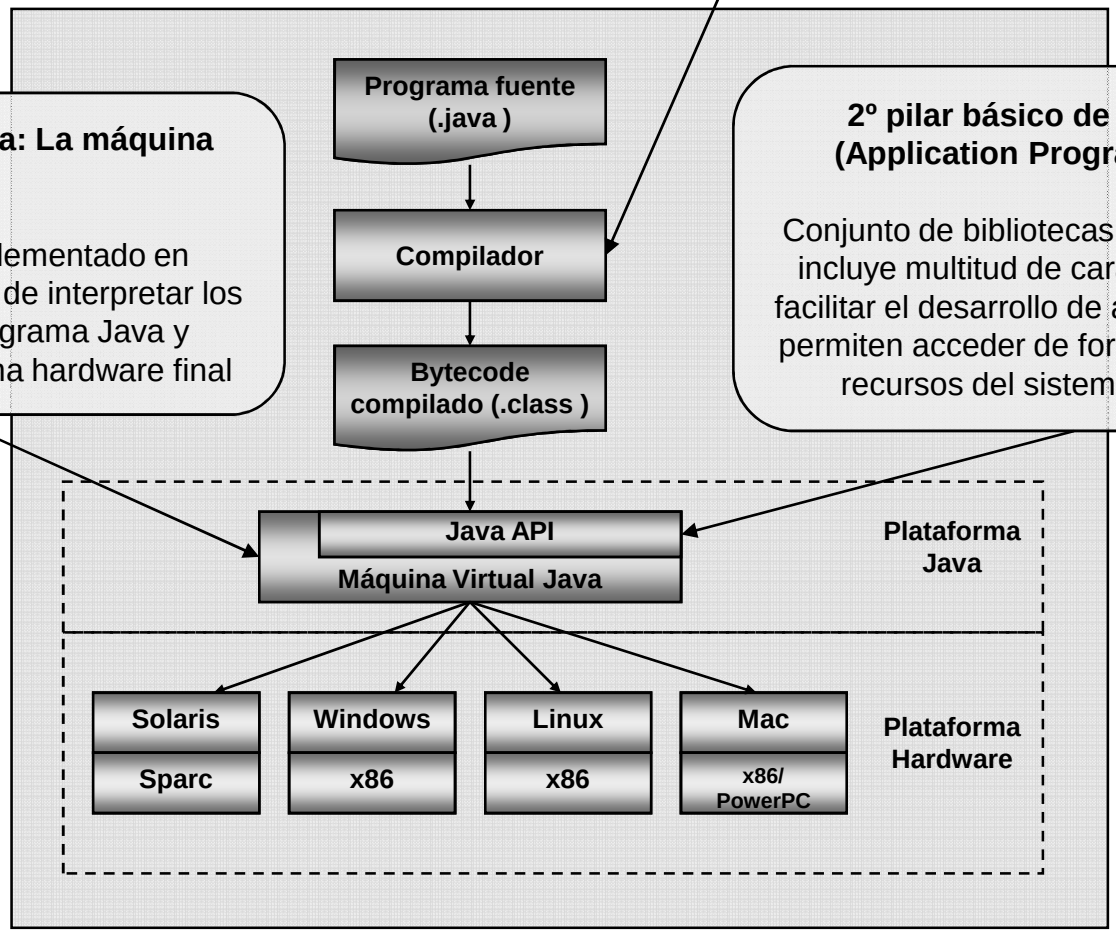
1er pilar básico de Java: El lenguaje

3º pilar básico de Java: La máquina virtual

Microprocesador implementado en software que se encarga de interpretar los ByteCodes de un programa Java y traducirlos a la plataforma hardware final

2º pilar básico de Java: El API (Application Program Interface)

Conjunto de bibliotecas de funciones que incluye multitud de características para facilitar el desarrollo de aplicaciones y que permiten acceder de forma estándar a los recursos del sistema subyacente



Lenguaje Java
Plataforma Java

- **Significado de los nombres**
  - **Java Platform: Especificación estándar de la plataforma Java**
  - **Java SDK o JDK: Plataforma Java que incluye herramientas de desarrollo**
  - **Java Runtime Environment (JRE): Plataforma Java que no incluye herramientas de desarrollo (pero permite ejecutar programas Java)**

- **Los caóticos números de versión**
  - **JDK 1.0 y JDK 1.1**
  - **J2SDK 1.2: el 2 es el número de la plataforma, el 1.2 la versión del lenguaje y el API. Siguieron las versiones 1.3 y 1.4**
  - **Java SE 5: la versión 1.5 pasa a ser la versión 5 (sin el .0) aunque no internamente (sigue siendo 1.5), la palabra Java aparece en el nombre y se abandona el incongruente 2. Siguió con la versión 6**

Lenguajes OO
Antecedentes Históricos

- **Curiosidades en los nombres de los lenguajes**
  - **El lenguaje C y sus sucesores:**
    - **CPL (Combined Programming Language)**
    - **BCPL (Basic CPL) → B → C**
    - **C++ (podría también ser D ó P)**
    - **C# (viene de C++++). Se lee "C sharp" ya que es el nombre del símbolo en inglés (sharp = afilado, agudo, listo, astuto).**
    - **En español "C almohadilla" no da esa impresión tan dinámica**
  - **El lenguaje Delphi**
    - **En la antigua Grecia cuando los griegos querían conocer el futuro iban a hablar con el Oráculo, para hablar con el había que ir a la ciudad de Delfos (Delphi en latín)**
    - **En la actualidad para "hablar" con la base de datos Oracle (literalmente Oráculo) puedes usar Delphi**
    - **La broma interna se convirtió en nombre definitivo de un producto que se iba a llamar Visual Pascal**
    - **La versión de Delphi para Linux se denominó Kylix. Un Kylix es una vasija griega, el nombre se escogió por ser un nombre similar a "Unix" o "Linux"**

Lenguaje Java
Historia

- **Principal rival: C# y .NET**
  - **En un primer intento Microsoft intentó dividir la plataforma Java con un lenguaje (J++) que tenía extensiones que únicamente funcionaban sobre Windows (típico ejemplo de “Embrace, extend and extinguish”)**
  - **Después de un largo pleito con Sun, Microsoft decidió abandonar J++ y construir su propia plataforma (.NET) y su propio lenguaje OO (C#)**
  - **Características:**
    - **.NET contempla el desarrollo de compiladores que traduzcan el código fuente a un código intermedio portable.**
    - **C# está basado en C++ pero eliminando aquellos aspectos que hacían a C++ propenso a errores**
    - **Esto me suena de algo…**
    - **Al ser un lenguaje más joven incluyó aspectos ausentes en Java y que facilitan su utilización (propiedades, delegados, etc.)**

Lenguaje Java
Historia

- **Java**
  - **Es una plataforma libre, liberada bajo licencia GPL en 2006**
  - **Sun controla la marca “Java” y también los tests de compatibilidad (TCK)**
  - **La evolución de Java se controla a través del JCP (Java Community Process) en el que participa Sun pero también otras compañías**
  - **Otras implementaciones: GNU GCJ (Compilador) y GNU Classpath (librerías) o el proyecto Harmony de Apache**
  - **Existen máquinas virtuales para múltiples sistemas (Windows, Linux, Mac, etc.)**
  - **Aunque Java es el principal lenguaje de la plataforma existen otros (Groovy, Scala, JRuby, etc.)**
  - **Los principales IDEs de Java son libres: NetBeans y Eclipse**

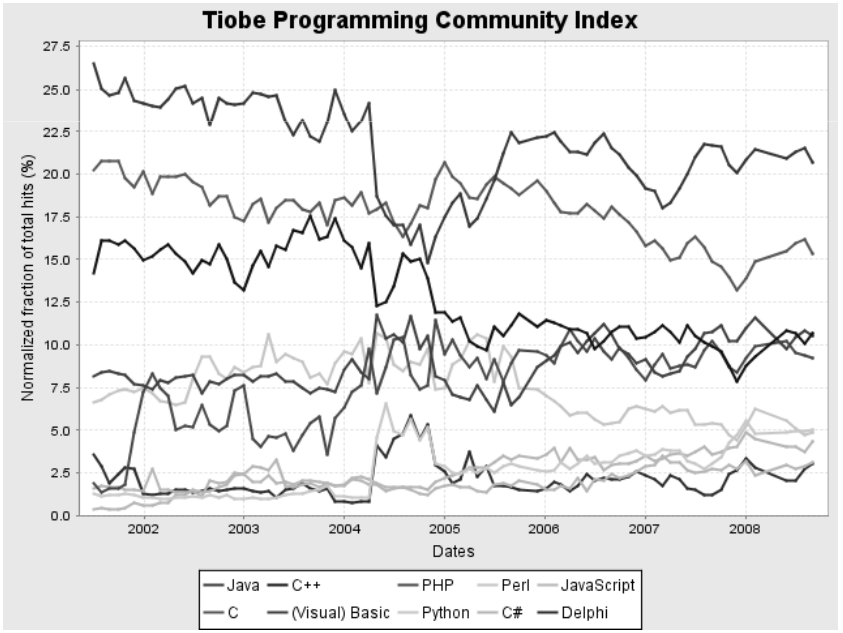
Lenguaje Java
Historia

- .NET
  - CLI (Common Language Infraestructure o máquina virtual) y C# están estandaricados por ECMA e ISO
  - No así otras partes (Windows Forms, ADO.NET, ASP.NET)
  - El proyecto Mono es la única alternative libre a .NET para otras plataformas (Linux)
  - Aunque C# es el principal lenguajes de la plataforma uno de los puntos fuertes de la misma es la existencia de múltiples lenguajes (Visual Basic.NET, Delphi.NET, etc.)
  - El principal IDE de .NET es software privativo de pago (Visual Studio) aunque existen versiones simplificadas gratuitas (Express) y libres (Mono Develop)

Lenguaje Java
Historia

- **TIOBE Programming Community Index (www.tiobe.com/tiobe_index)**

Position Sep 2008	Position Sep 2007	Delta in Position	Programming Language	Ratings Sep 2008
1	1	=	Java	20.715%
2	2	=	C	15.379%
3	5	⇑	C++	10.716%
4	3	⇓	(Visual) Basic	10.490%
5	4	⇓	PHP	9.243%
6	8	⇑	Python	5.012%
7	6	⇓	Perl	4.841%
8	7	⇓	C#	4.334%
9	9	=	JavaScript	3.130%
10	14	⇑⇑⇑⇑	Delphi	3.055%



Lenguaje Java
Plataforma Java

- **Otras versiones de la plataforma Java**
  - **Java Enterprise Edition o Java EE.**
    - **Esta plataforma puede considerarse como un superconjunto de la plataforma Java 2, Standard Edition. La tecnología que se incluye en esta nueva plataforma es esencialmente un conjunto de extensiones centradas en los servidores de red: Enterprise Java Beans (EJB), Servlets, Java Server Pages (JSP), etc.**
  - **Java Micro Edition o Java ME.**
    - **Se trata de un entorno de ejecución Java altamente optimizado destinado a ser utilizado en pequeños dispositivos como tarjetas inteligentes, teléfonos móviles o set-top boxes.**