MASTER EN MODELIZACIÓN MATEMÁTICA, ESTADÍSTICA Y COMPUTACIÓN
2020-2021

Curso: Bases de datos y programación orientada a objetos
Parte POO

Introducción

Nuestro objetivo:

1. Entender la Programación Orientada a Objetos.
2. Aplicarla al acceso a Bases de Datos

Objetivo 1 – Aprenderemos un lenguaje para entender los conceptos y poder aplicarlos (nos extenderemos un poco inicialmente sobre la elección del mismo)

Objetivo 2 – Con programación Orientada a Objetos es casi trivial. Una vez sabemos cómo usar objetos, vale con servirnos de los adecuados.

Antes de nada…

…por si es Python el lenguaje más usado por los alumnos de la asignatura

Oct 2020	Oct 2019	Change	Programming Language	Ratings	Change
1	2	^	C	16.95%	+0.77%
2	1	v	Java	12.56%	-4.32%
3	3		Python	11.28%	+2.19%
4	4		C++	6.94%	+0.71%
5	5		C#	4.16%	+0.30%

https://www.tiobe.com/tiobe-index/

Python es un lenguaje exitoso en ciencia y tecnología, ¿no podemos utilizarlo en la asignatura?

Year	Winner
2019	C
2018	Python
2017	C
2016	Go
2015	Java
2014	JavaScript
2013	Transact-SQL
2012	Objective-C
2011	Objective-C
2010	Python
2009	Go
2008	C
2007	Python
2006	Ruby
2005	Java
2004	PHP
2003	C++

Python tiene mucho de Orientación a Objetos, pero… es “error-prone”…

En realidad se usa habitualmente “sin saber” de Orientación a Objetos porque sólo se utiliza como “aglutinador” de librerías (que normalmente se han desarrollado en C/C++ (Python es demasiado lento))

IDEAS GENERALES DE
Programación Orientada a Objetos

¿Qué es la programación Orientada a Objetos?

Un modo de desarrollar software en el que

Nos focalizamos en escribir “piezas” razonablemente reducidas de modo que podemos “controlarlas” con facilidad (no sobrepasan nuestra capacidad mental para tener en cuenta detalles o estructuras)

Independiente del objetivo final -la aplicación informática-, estas piezas se plantean pensando en su propia coherencia (en este sentido representarán objetos reales o conceptos claramente definidos). Obviamente no es “del todo” cierto ese “independientemente”, pero sí lo es en buena medida (pensemos en “Persona” para un hospital o para una universidad… no son lo mismo).

Los objetos “se apoyan” en otros objetos para todo aquello que resulta complejo dentro de su planteamiento (pensemos en “Persona” en el ámbito universitario, y su “ExpedienteAcadémico”).

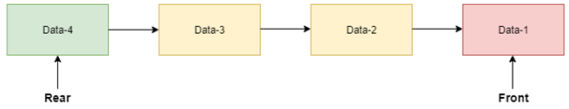
La aplicación informática, como otro objeto cualquiera, será una composición también razonablemente sencilla en el que las complejidades se delegan en otros objetos.

¿Por qué surge la programación Orientada a Objetos?

Esta idea de “tipo complejo” ya venía en cierto modo practicándose desde antes de definirse la Orientación a Objetos. Un punto concreto en donde podía apreciarse bien era en el planteamiento de lo que se conoce como “Tipos Abstractos”.

Los Tipos Abstractos son estructuras de datos que se definen por las operaciones que se pueden hacer sobre ellos independientemente de cómo se concreten en la memoria del ordenador (p.ej. una “cola” o “FIFO” con las operaciones “enqueue” y “dequeue”) .

Limited number of data we can store



We can create unlimited (ignoring memory restriction) node and append to the queue.

Los “buenos programadores” estructuraban su código de un modo similar a lo que hoy es la definición de clases, y la observación de que su formalización daba lugar a nuevos conceptos y capacidades de gran potencia estructural, llevó a formalizarse y considerarse un nuevo paradigma.

¿Cómo funciona un programa Orientado a Objetos?

En definitiva un programa orientado a objetos es una colección de objetos colaborando entre ellos con un determinado fin o proporcionando unas determinadas capacidades.

Los objetos se comunican entre sí “enviándose mensajes” (“pidiendo servicios”, “dando ordenes”… como queramos verlo.

La terminología “enviar mensajes” puede concretarse de diferentes maneras: desde un modo estricto, es decir, enviar mensajes con estructura predeterminada a través de un canal de comunicaciones, hasta ejecutar llamadas a subrutinas o compartir datos en un área de memoria común.

En todo caso se trata de flujos de información que estarán determinados/definidos por sus “interfaces” (la parte que los objetos muestran al exterior).

¿Cómo se escribe un programa Orientado a Objetos?

Escribiendo (muchas) clases de las que se instancian Objetos.

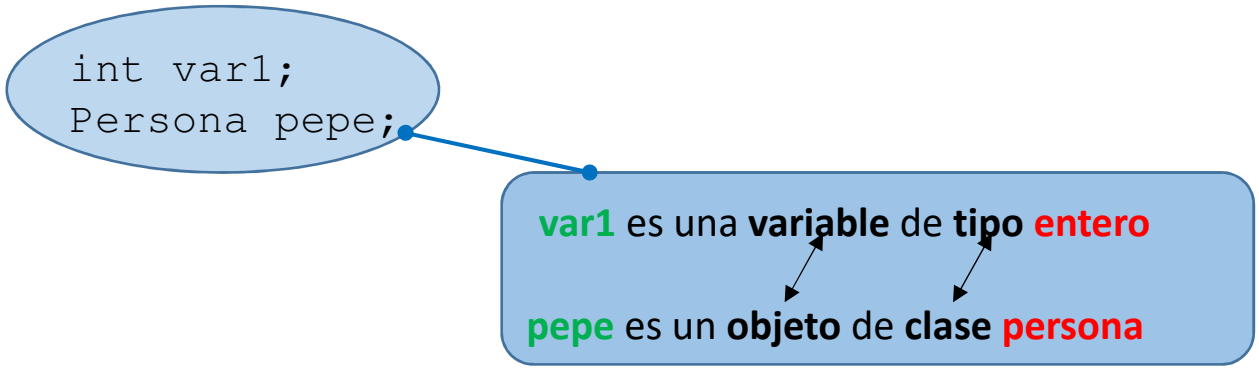
En la terminología original se hablaba de Tipos y Objetos y hoy hablamos de Clases y Objetos. Desde un punto de vista "mecánico", los Objetos son realizaciones concretas (instancias) de "Tipos complejos", del mismo modo que las variables son realizaciones concretas de tipos simples (en terminología Java, "tipos primitivos").

```
int       contador, índice, resultado;(*)
Persona   pedro, maría;
```

Para programar con Objetos basta con hacerlo en un entorno que los proporcione (por ejemplo con javaScript dentro de un navegador web). Esto es lo que se denomina programación "basada" en objetos (actualmente ampliada como programación "basada en prototipos"). Si además tenemos la capacidad de definir nuevos "tipos complejos" (nuevas Clases), hablamos de programación Orientada a Objetos.

(*) sobre la inconveniencia de usar no-ascii en programación

Clase es a **tipo** como **objeto** es a **variable**



Una clase es un “tipo complejo”; una agrupación de variables (constantes), objetos, e incluso código que puede actuar sobre sus propios elementos u otros.

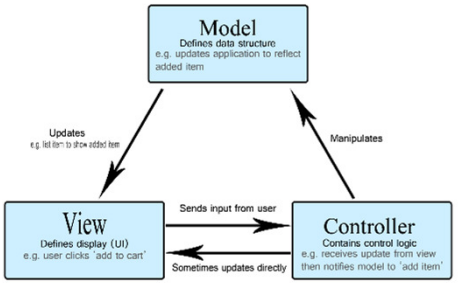
Un objeto es una cápsula (de memoria de ordenador) que tiene un “estado” (determinado por los valores de sus variables y el estado de sus objetos) así como un comportamiento (definido por el código que encierra).

Básicamente lo que venían haciendo los buenos programadores antes de que se formalizara el concepto… …pero la formalización abrió un mundo de posibilidades.

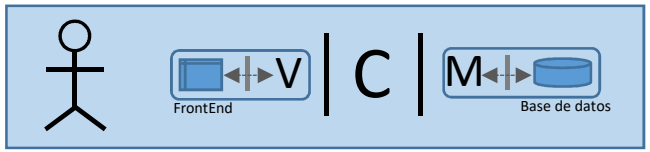
¿Cómo se estructura un programa Orientado a Objetos?

La Orientación a Objetos potenció enormemente la arquitectura de aplicaciones. Independientemente de estructuraciones más específicas, un concepto básico es el de la estructura en capas (tiers), que en su versión básica consiste en tres capas, lo que se conoce como modelo MVC (Model-View-Controller) o ECB (Entity-Control-Boundary)

Las capas "exteriores" pueden desdoblarse para dar lugar a un modelo de hasta 5 capas, donde ahora las más externas son aplicaciones ligeramente acopladas al núcleo de 3 capas.



La parte M de esta estructura en capas, enlaza directamente con lo que se trata en este curso, y además nos permite introducir un "cierto" paralelismo entre las clases M y las tablas de una base de datos.



Sobre la elección del lenguaje (Java) y lo que nos va a proporcionar

(con un poco de acento en las matemáticas)

La Programación ES y VIENE DE las Matemáticas… (como parte de la computación)
Del concepto de cálculo efectivo a una

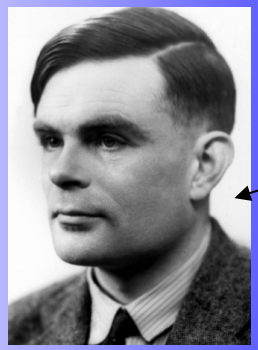
Clasificación general de lenguajes

Euclides (Método axiomático), Aristóteles(Lógica formal), Muhammad ibn Musa Al'Khowarizmi (Algorítmo)…

Sin olvidar las funciones recursivas de Herbrand-Gödel"

..1900..
Frege
Cantor
Russell-Whitehead
HILBERT
Zermelo-Fankl
GÖDEL
TURING - CHURCH
1938

Máquina de Turing



Alan Turing

Lenguajes

Alonzo Church

Cálculo Lambda

Imperativos
- Fortran
- Cobol
- Pascal
- C
- …

Funcionales
- Lisp
- Scheme
- ML
- Hope
- CLOS
- …

Lógicos
- Prolog
- …

O/B objetos
- Object Pascal
- C++
- Java
- …

Frameworks
- Ruby on Rails
- …

```
let rec long = function
        |[] -> 0
        |x::xs -> 1 + long xs;;
let rec ordenar = function
        |[] -> []
        |x::xs -> insertar x (ordenar xs)
and insertar e = function
        |[] -> [e]
        |x::xs -> if x > e
        then e::x::xs
        else x::(insertar e xs);;
```

Ejemplo OCaml

1928 Hilbert-Ackermann: Principles of Mathematical Logic
1931 Kurt Godel: "On Formally Undecidable Propositions of Principia Mathematica and Related Systems I"
1936-38 Alan M.Turing: "On Computable Numbers, with an Application to the Entscheidungsproblem" (Entscheidung=decisión)

Paradigmas en programación …

Históricamente –y sobre el papel– comenzó con Ada Byron (1843) y la Máquina Analítica de Charles Babbage.

En realidad, una vez construidos las primeras computadoras eléctricas/electrónicas, empezó totalmente ligada al hardware como configuración del mismo mediante cables

El primer "lenguaje" –el A– se debe a Grace Murray Hopper (1952).

Un primer hito fue el concepto de "subrutina"

El primer "paradigma" importante la "Programación Estructurada"

El segundo la "Programación Orientada a Objetos"

La idea de "arquitectura software" surge con la Programación Estructurada, pero es con la Programación Orientada a Objetos como adquiere un gran desarrollo. La POO facilita la definición de "Patrones" a todos los niveles, de modo que se han sucedido varios "paradigmas" de arquitectura. Otros paradigmas pueden tener éxito (p.ej. AOP -Aspect Oriented Programming), pero no son indiscutibles como ya lo es la POO.
Otra cosa son las arquitecturas de desarrollo. Actualmente despunta la aproximación SOA –Service Oriented Architecture– (facilitando SaaS) y se sigue evolucionando con soluciones a las dificultades que presenta en determinadas condiciones.

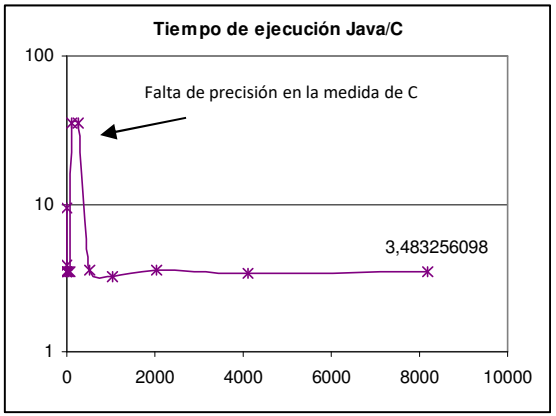
¿Java para cómputo intensivo?... Una experiencia concreta

Un ejecutable desarrollado con Java puede ser algo más lento que con otros lenguajes pero...

Ejemplo peor caso Java vs. C (14ago08)
(cálculo de PI por MonteCarlo)

Experimento a partir del código tomado de http://husnusensoy.blogspot.com/2006/06/c-vs-java-in-number-crunching.html

- Comparación del tiempo de ejecución



La relación de tiempo de ejecución es del orden de 3,5 a favor de C

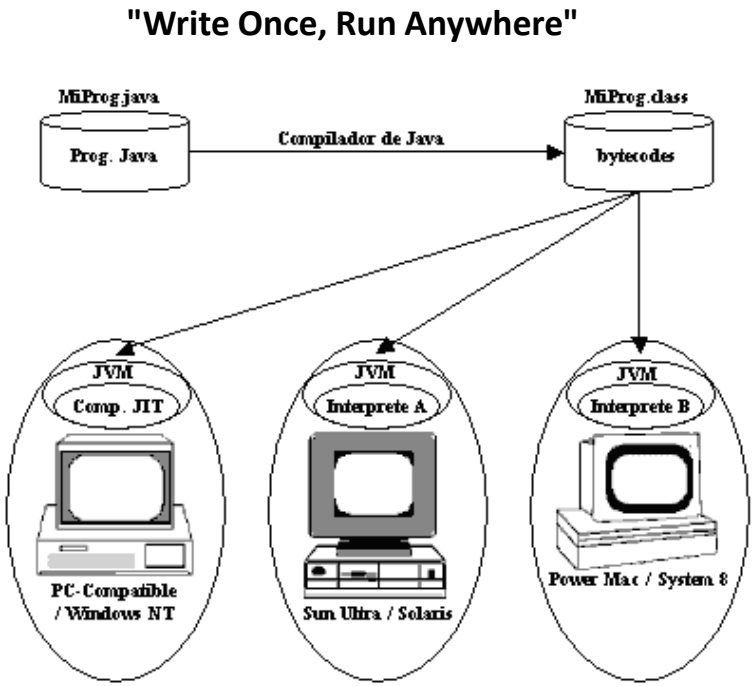
- Comparación del tiempo de preparación del experimento

JAVA:
- copiar, pegar, compilar, ejecutar y **listo en unos segundos**.

C:
- copiar, pegar, compilar, errores… (no coincide exactamente el lenguaje)
- corregir fuente, compilar, ejecutar, errores… (la arquitectura de la máquina no es la adecuada)
- corregir fuente, compilar, ejecutar, se observar falta de resolución de la función “time”,
- ir a la bibliografía para resolver el tema, no encontrar solución...
- replantear con iteraciones para obtener tiempos mayores…
- cambiar fuente compilar, ejecutar… errores de apuntadores (falta de práctica de un “ex” de C)
- corregir fuente, compilar, ejecutar… errores de violación de segmentos
- corregir fuente, compilar, ejecutar y… **listo en una hora**.

La relación de tiempo de preparación ha sido de 120 a favor de Java

En la máquina Virtual está el “secreto”… mejor cada día, y quien quiera puede innovar.



Proprietary/closed source implementations
* Hewlett-Packard's Java for HP-UX, OpenVMS, Tru64 and Reliant (Tandem) UNIX platforms
* J9 VM from IBM, for AIX, Linux, MVS, OS/400, Pocket PC, z/OS
* Mac OS Runtime for Java (MRJ) from Apple Inc.
* JRockit from BEA Systems acquired by Oracle Corporation
* Oracle JVM (also known as "JServer" and as "OJVM") from Oracle Corporation
* Microsoft Java Virtual Machine (MS JVM) from Microsoft
* PERC from Aonix is a real time Java for embedded
* JBed from Esmertec is an embedded Java with multimedia capabilities
* JBlend from Aplix is a Java ME implementation
* Excelsior JET (with AOT compiler)

Lesser-known proprietary JVMs
* Blackdown Java (port of Sun JVM)
* CVM
* Gemstone Gemfire JVM - modified for J2EE features
* Golden Code Development (EComStation and OS/2 port of Java RTE and SDK for J2SE v1.4.1_07)
* Tao Group's intent
* Novell, Inc.
* NSIcom CrE-ME
* HP ChaiVM and MicrochaiVM
* MicroJvm from Industrial Software Technology (running of wide rande of microcontrollers 8/16/32-bit)

Free/open source implementations
* AegisVM
* Apache Harmony
* CACAO
* IcedTea
* IKVM.NET
* Jamiga
* JamVM
* Jaos
* JC
* Jikes RVM
* JNode
* JOP
* Juice
* Jupiter JVM
* JX (operating system)
* Kaffe
* leJOS
* Mika VM
* Mysaifu JVM
* NanoVM
* SableVM
* SuperWaba
* TinyVM
* JESSICA (Java-Enabled Single-System-Image Computing Architecture)
* Squawk virtual machine (Sun JVM for embedded system and small devices)
* Sun Microsystems' HotSpot
* VMkit of Low Level Virtual Machine
* Wonka VM
* Xam

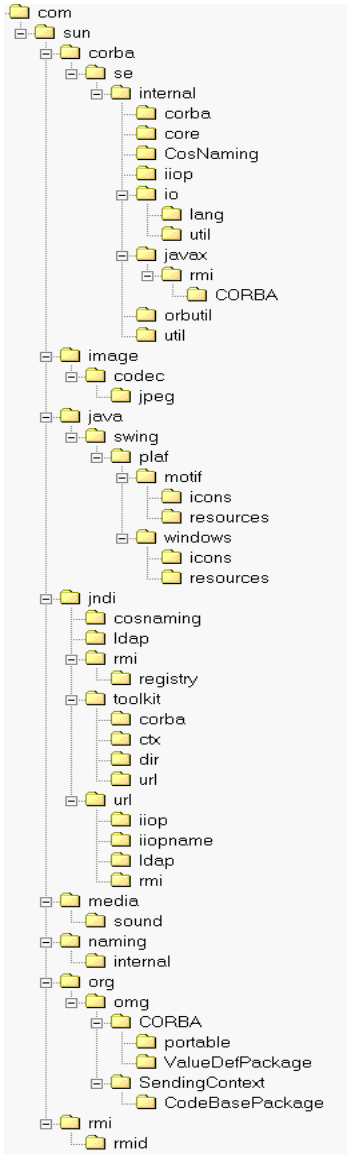
WIKIPEDIA
The Free Encyclopedia

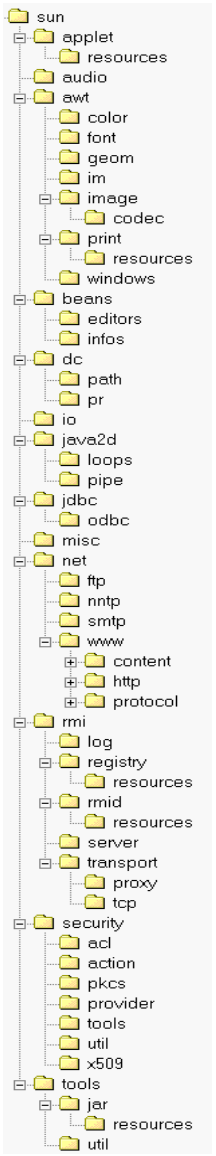
- Una idea novedosa, pero no del todo: cierta similitud con los lenguajes con código intermedio.
- Sí fue novedoso el enfoque de emulador de máquina (y la compilación JIT).
- Ventajas:
  - se pueden incluir con facilidad técnicas que en un diseño hardware pueden resultar prohibitivas por su complejidad técnica,
  - la posibilidad de evolución es mucho más sencilla al no requerir cambios de hardware
  - permite utilizar las "plataformas" existentes sin implicar una ruptura con los sistemas actuales (existe la máquina real pero…).
- **el diseño es público y la "implementación" es privada** (**especificaciones técnicas que debe complir toda JVM.**).
  - Distintos comportamientos en términos de velocidad y uso de memoria

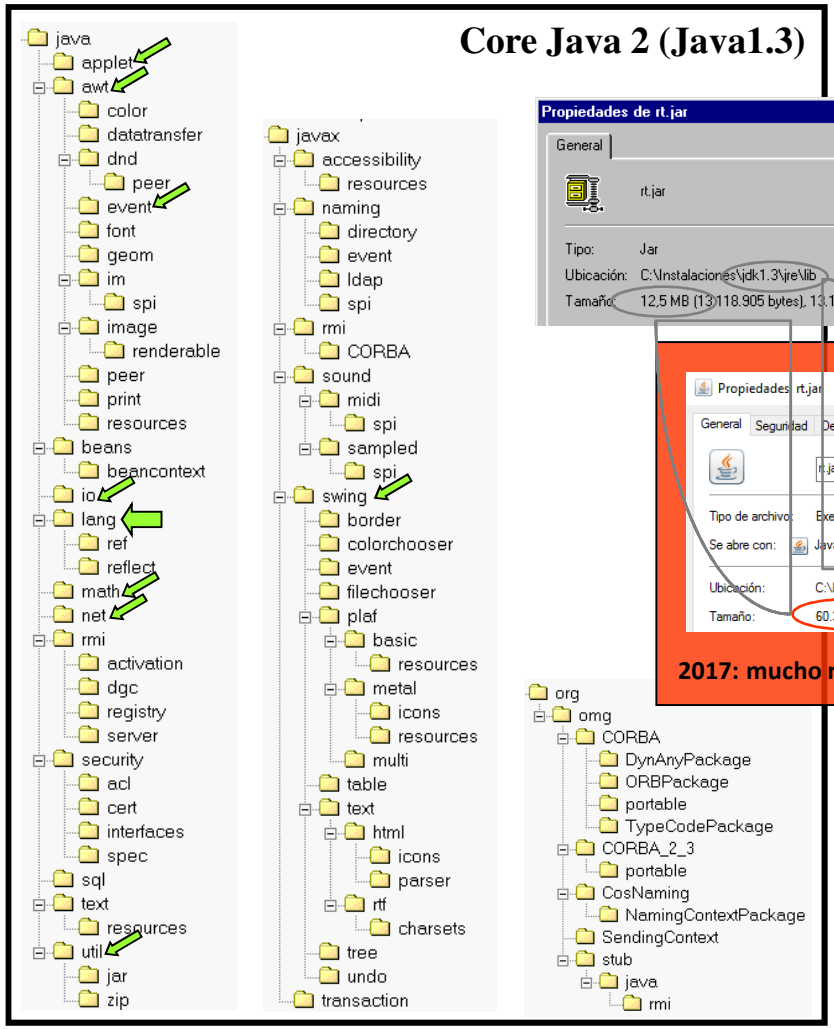
Una gran ventaja: La biblioteca de ejecución de Java

Estructura del contenido de C:\...\jdk1.3\jre\lib\rt.jar
Mucho mayor en 1.8

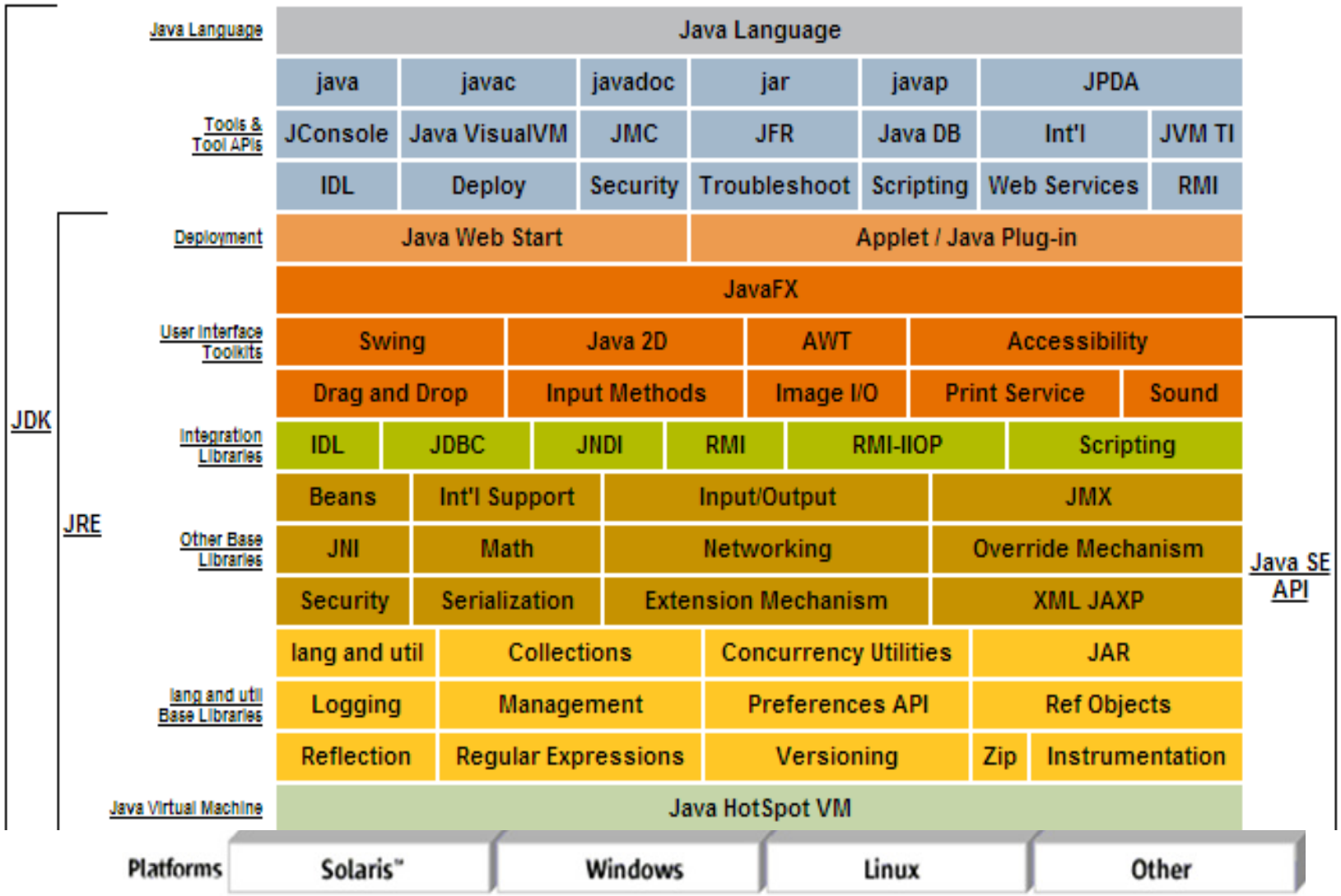
Ojo: 1.9 ya no tiene rt.jar
(pero la biblioteca se estructura igual)



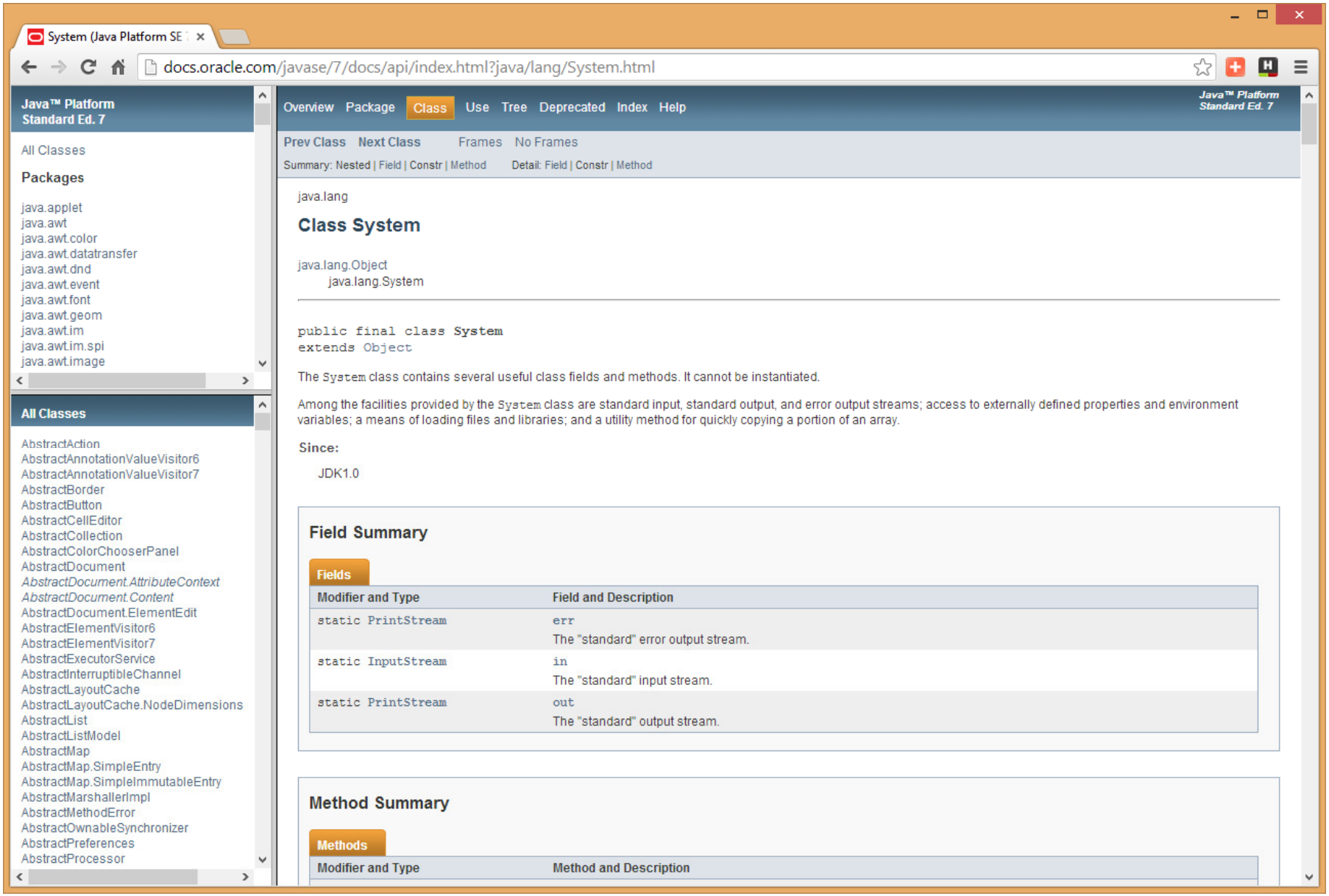




La biblioteca vista de un modo más estructurado



Otra gran ventaja: la documentación



Más ventajas: copiar, copiar y copiar.

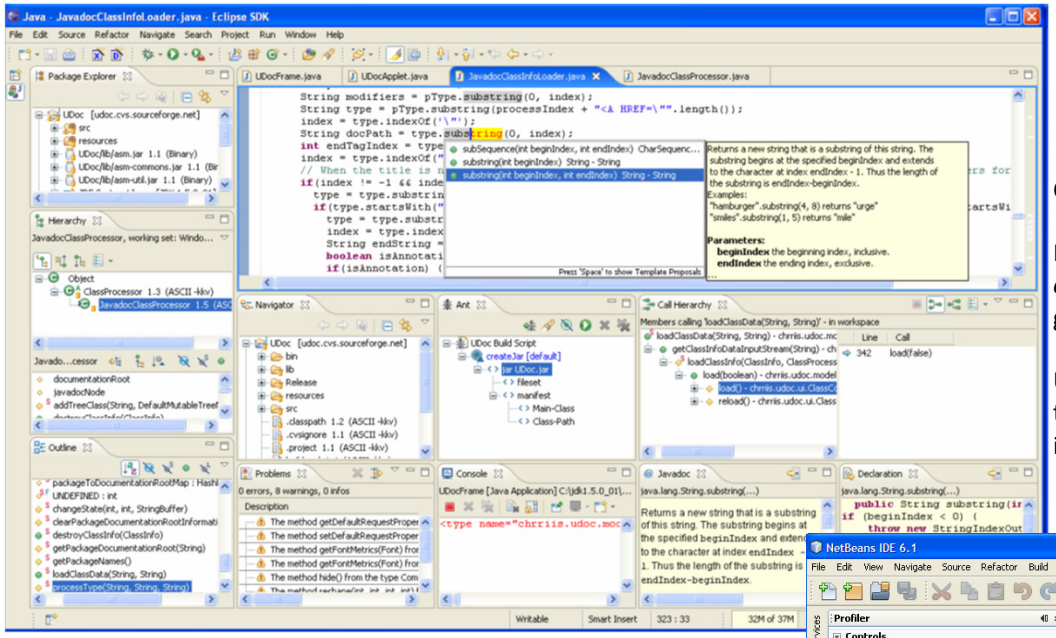
Los programas ejecutables Java. Desensamblado de codebytes y decompilación


```
C:>javap -c HolaMundo
Compiled from "HolaMundo.java"
public class HolaMundo extends java.lang.Object{
public HolaMundo();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

public static void main(java.lang.String[]);
  Code:
   0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
   3:   ldc     #3; //String Hola, mundo
   5:   invokevirtual   #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
   8:   return

}
```

Decompilación: probar con Java Optimize and Decompile Environment (JODE)

"Ofuscacion"

IDEs (Integrated Development Environments) para desarrollo en Java

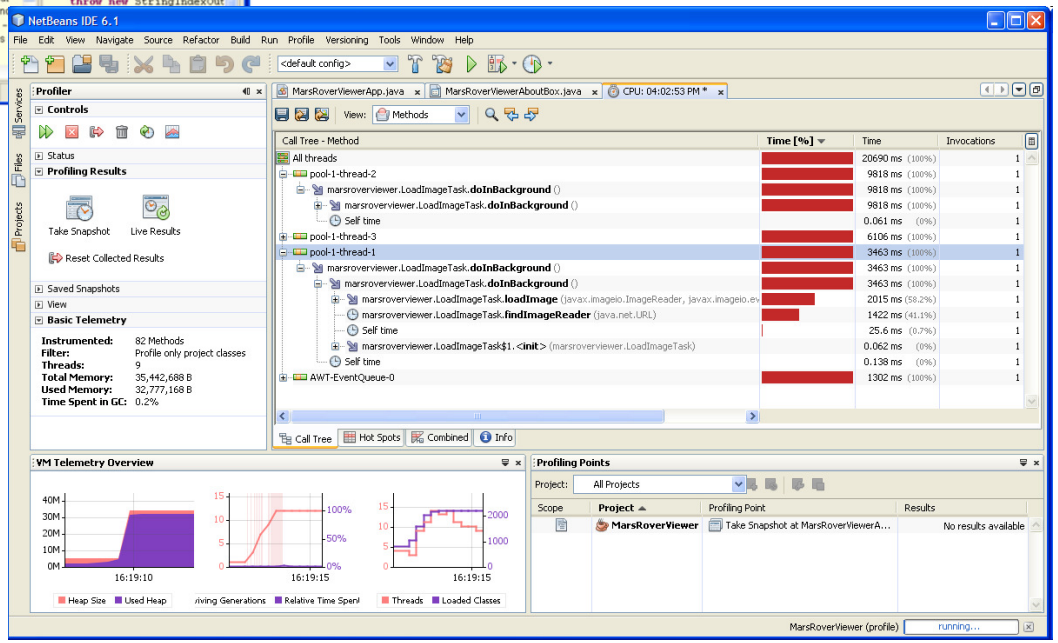


ECLIPSE (.org)

Comunidad de desarrollo en código abierto

Proyectos enfocados al desarrollo de una plataforma de *marcos extensibles*, *herramientas* y ejecutables para construir, implantar y gestionar software a lo largo de todo su *ciclo de vida*.

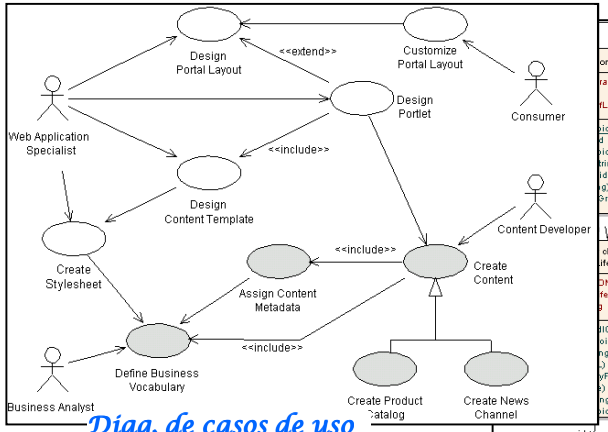
Un "vibrante" y "amplio" ecosistema de grandes fabricantes de tecnología, innovadoras start-ups, universidades, instituciones de investigación y particulares.



NETBEANS(.org)

Un IDE de código abierto gratis para desarrolladores de software. Proporciona todas las *herramientas* para crear aplicaciones profesionales de sobremesa, de empresa, web y móviles, con Java, C/C++, y Ruby. NetBeans es fácil de instalar y usar de inmediato, y corre en numerosas plataformas incluyendo Windows, Linux, Mac OS X y Solaris.

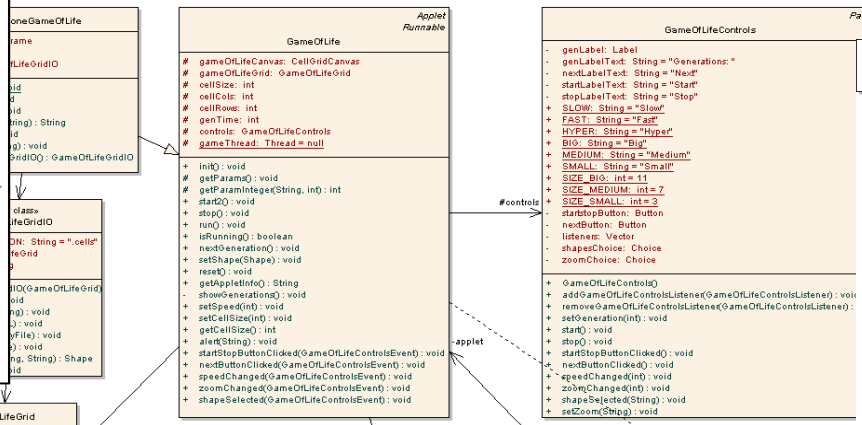
¿Desarrollo en UML?

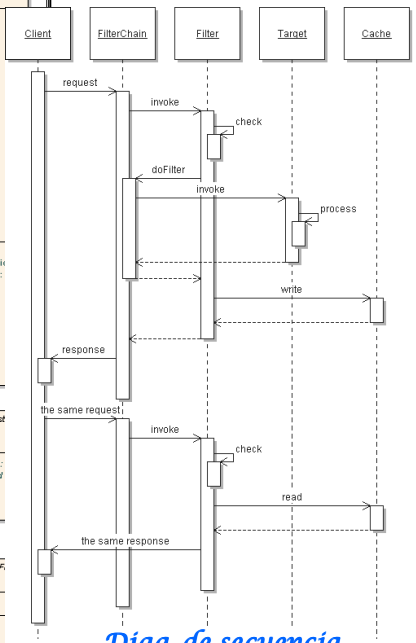


Diag. de casos de uso

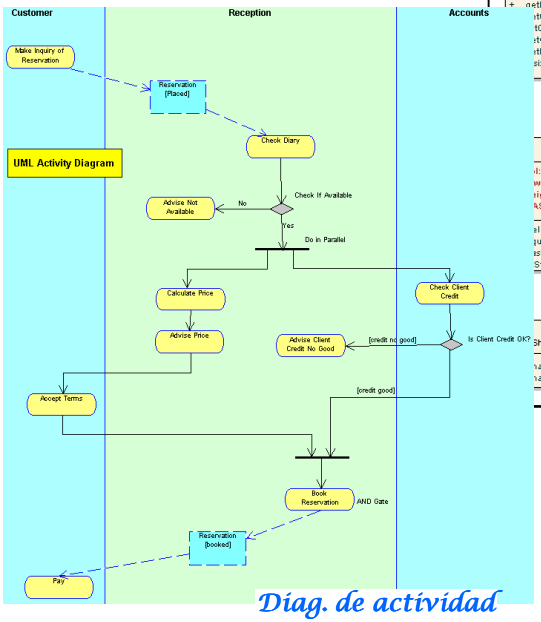
Desarrollo mediante modelado: UML

Diag. de clases

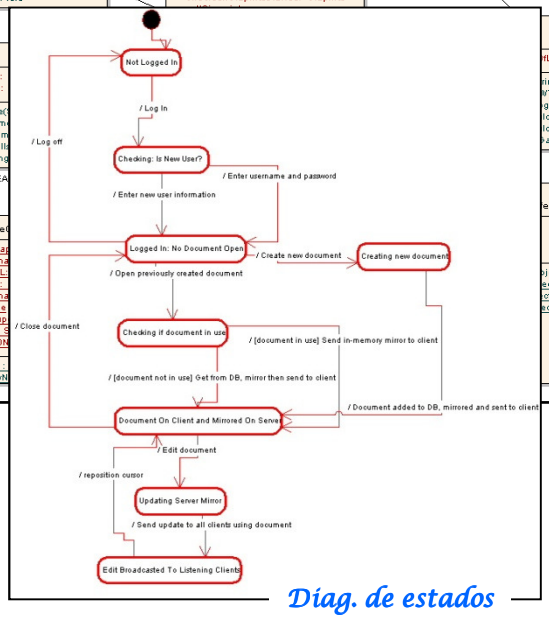




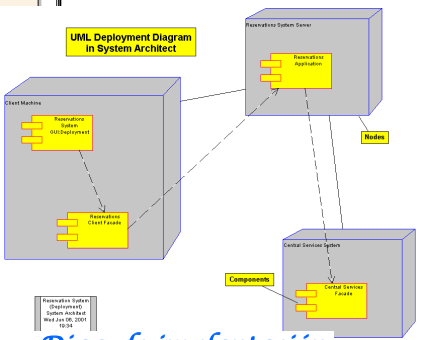
Diag. de secuencia



Diag. de actividad



Diag. de estados



Diag. de implantación

Comenzamos, pero…¿Qué es mejor, simple o complejo?

¡HOLA MUNDO!

COBOL

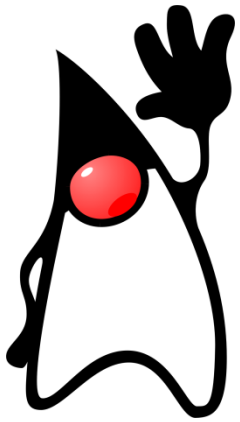
```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY 'Hello, world!'.
    STOP RUN.
```

FORTRAN

```
program hello
    write(*,*) 'Hello, world!'
end program hello
```

Pascal

```
program HelloWorld;

begin
  WriteLn('Hello, world!');
end.
```



Lisp

```
(print "Hello, world!")
```

Ocaml

```
print_endline "Hello, world!"
```

Haskell

```
main = putStrLn "Hello, world!"
```

C

```
#include <stdio.h>

int main(void)
{
    puts("Hello, world!");
}
```

Clojure

```
(println "Hello, world!")
```

C++

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
}
```

C#

```
using System;
class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Hello, world!");
    }
}
```

JAVA

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Python

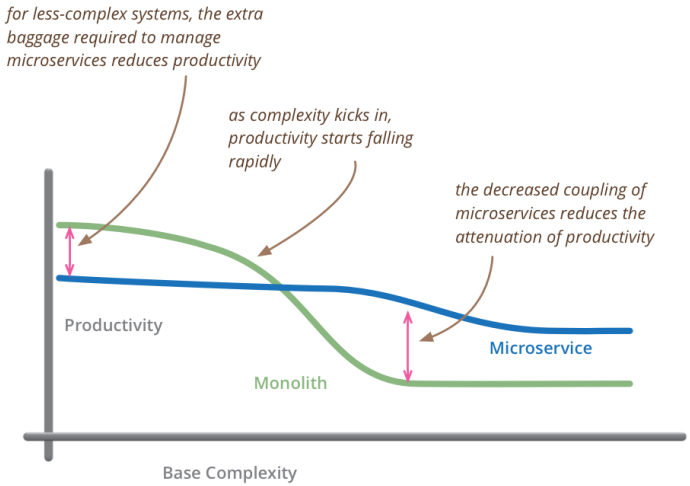
```
print "Hello, world!"
```

Javascript

```
console.log('Hello, world!');
```

SCALA

```
object HelloWorld extends App {
  println("Hello, world!")
}
```



Se refiere a otro tema (arquitectura basada en microservicios), pero es igualmente aplicable a lo que nos ocupa

Microservices: productivity versus base complexity (source: Martin Fowler; http://martinfowler.com/bliki/MicroservicePremium.html)

JAVA

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Ejecutar!