MASTER EN MODELIZACIÓN MATEMÁTICA, ESTADÍSTICA Y COMPUTACIÓN
2019-2020

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

Introducción y puesta en marcha

La Programación ES y VIENE DE las Matemáticas… (como parte de la computación)

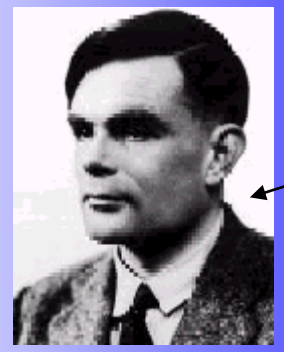
Del concepto de cálculo efectivo a una

Clasificación general de lenguajes

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

Sin olvidar las funciones recursivas de Herbrand-Gödel"

Máquina de Turing



1928
HILBERT
GÖDEL
1938
Lenguajes

Cálculo Lambda

Alan Turing

Alonzo Church

Imperativos
- Fortran
- Cobol
- Pascal
- C
- …

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

Lógicos
- Prolog
- …

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

Frameworks
- Ruby on Rails
- …

Ejemplo OCaml

```
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);;
```

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)

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"

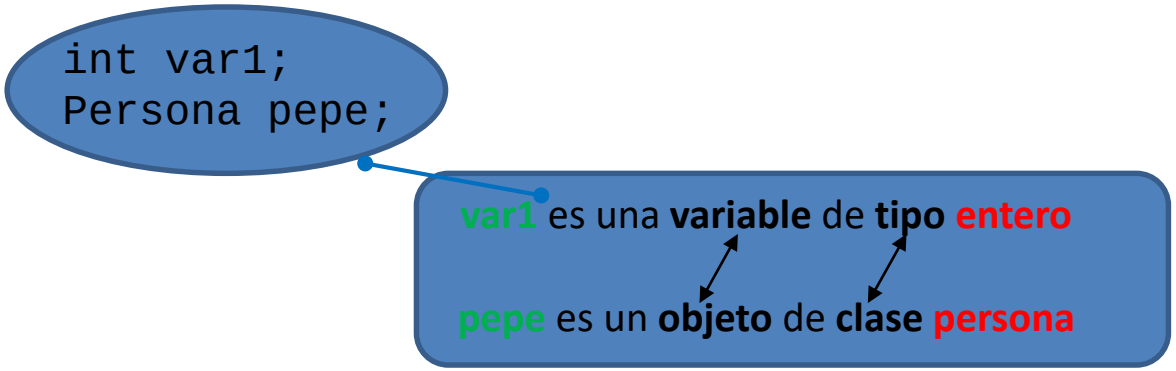
El segundo la "Programación Orientada a Objetos"

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.

Objetos? …

Una MÍNIMA idea de lo que es un “objeto” hasta que nos detengamos en ello…

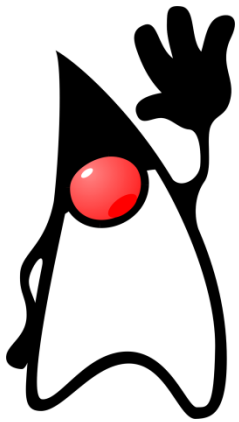
Clase es a **tipo** como **objeto** es a **variable**



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.



¡HOLA MUNDO!

¿Qué es mejor, simple o complejo?

COBOL

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

FORTRAN

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

Lisp

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

Pascal

```
program HelloWorld;

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

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

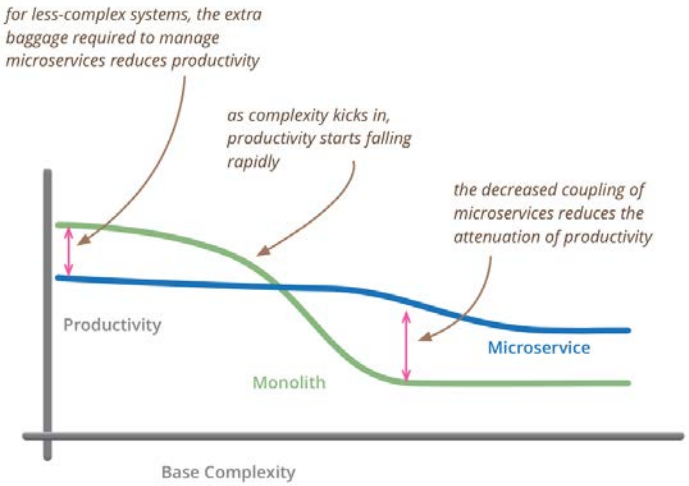
```
print "Hello, world!"
```

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!

JAVA? POR QUÉ ES INTERESANTE INCLUSO EN MATEMÁTICAS

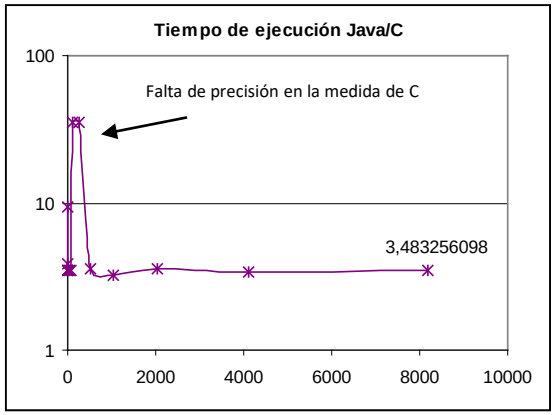
¿Java para cómputo intensivo?... Una experiencia concreta

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

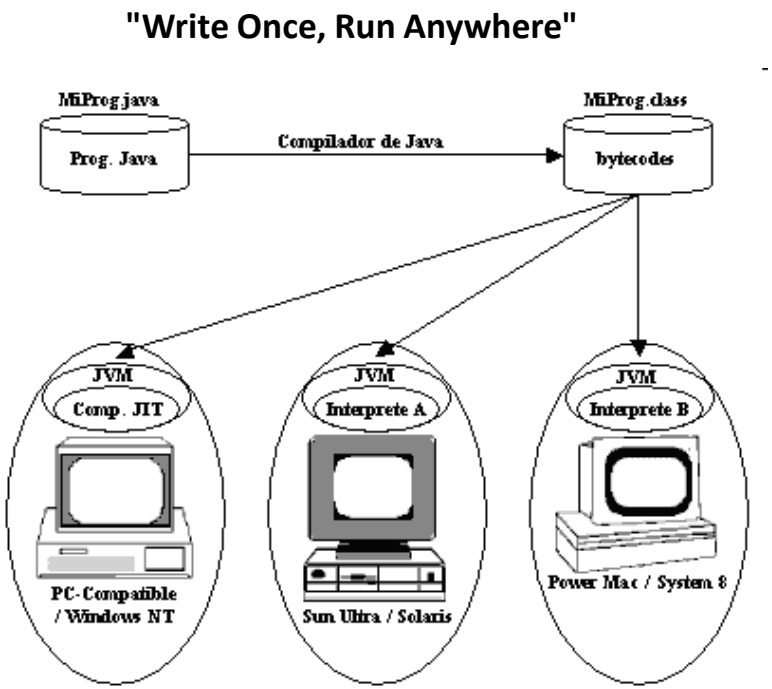
- Comparación del tiempo de preparación del experimento

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

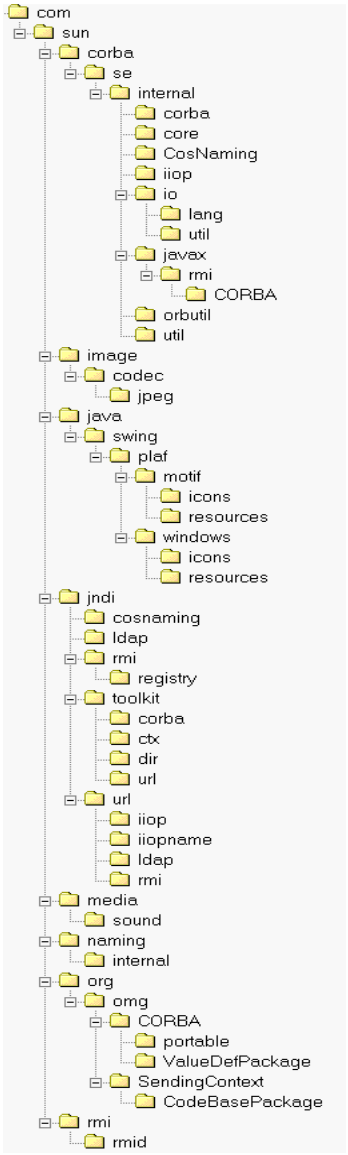
* JESSICA (Java-Enabled Single-System-Image Computing Architecture)
* Squawk virtual machine (Sun JVM for embedded system and small devices)
* Sun Microsystems' HotSpot
* VMkit of Low Level Virtual Machine
* Wonka VM
* Xam

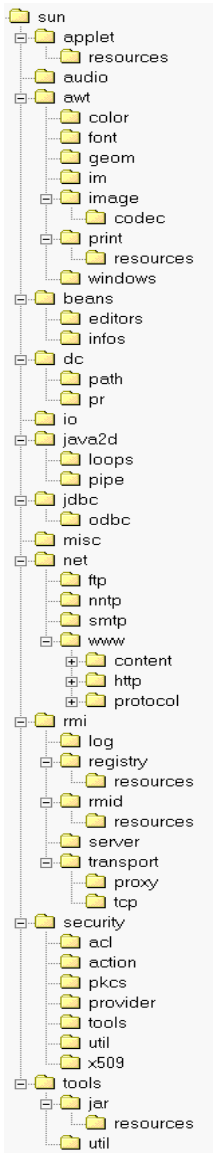
- Una idea novedosa, pero no del todo: cierta similitud con los lenguajes con código intermedio.
- Sí fue novedoso el enfoque de emulador de máquina (y la compilación JIT).
- Ventajas:
  - se pueden incluir con facilidad técnicas que en un diseño hardware pueden resultar prohibitivas por su complejidad 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

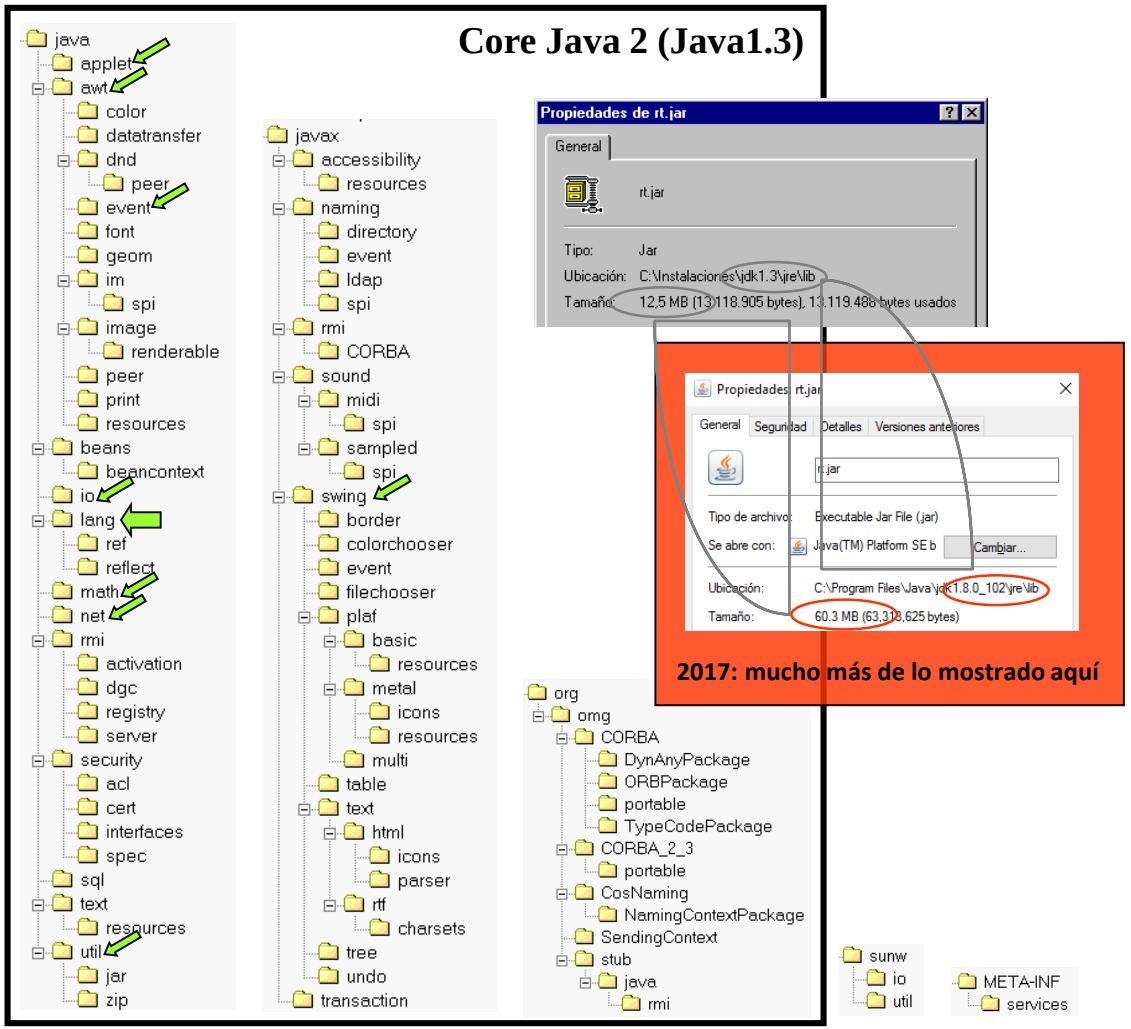
Una gran ventaja: La biblioteca de ejecución de Java

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

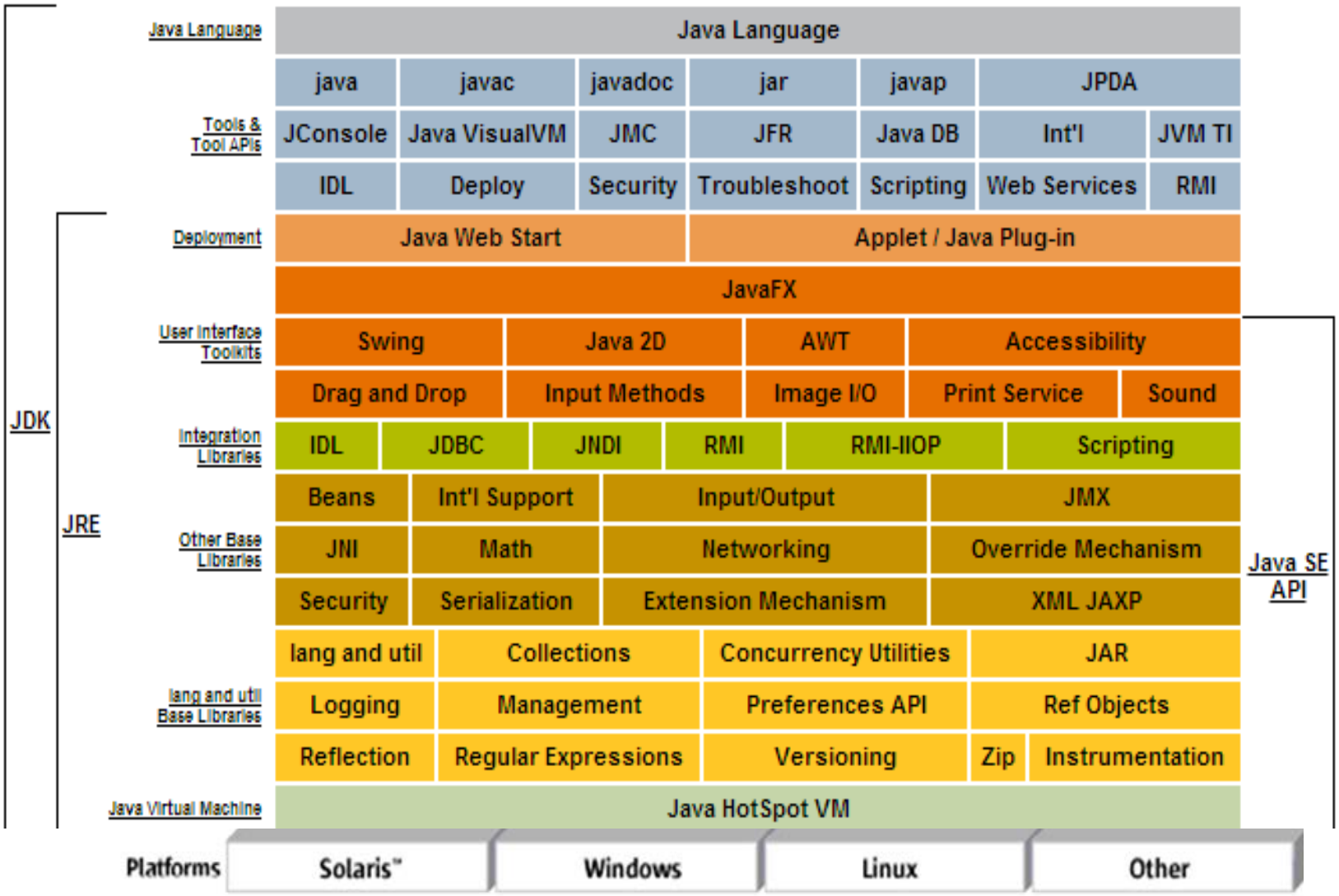
Ojo: 1.9 ya no tiene rt.jar
(pero la biblioteca se estructura igual)



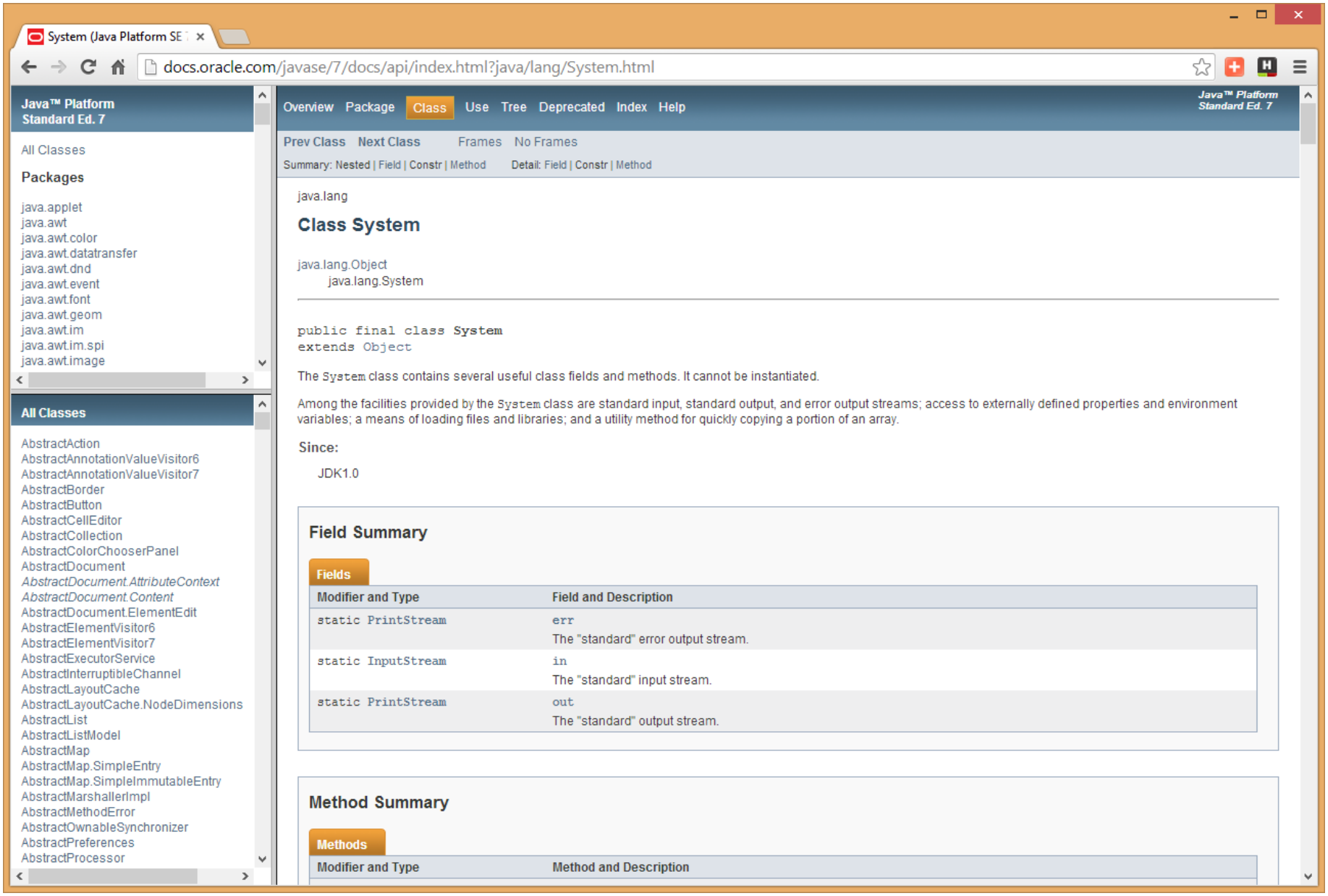




La biblioteca vista de un modo más estructurado



Otra gran ventaja: la documentación



Más ventajas: copiar, copiar y copiar.

Los programas ejecutables Java. Desensamblado de codebytes y decompilación

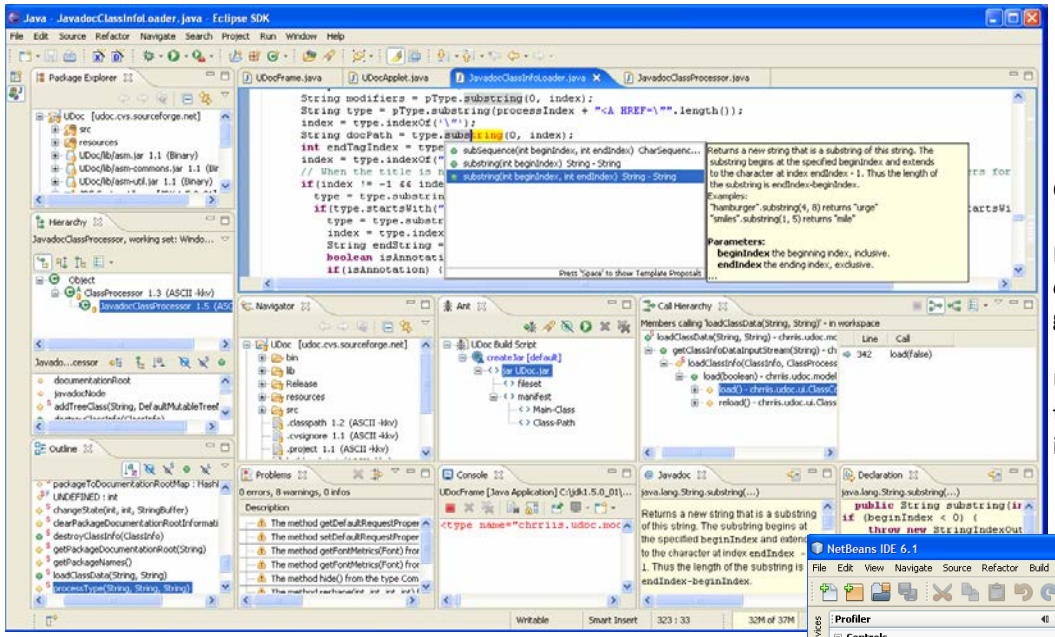

```
C:>javap -c HolaMundo
Compiled from "HolaMundo.java"
public class HolaMundo extends java.lang.Object{
public HolaMundo();
  Code:
   0:   aload_0
   1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
   4:   return

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"

¿CÓMO PUEDO DESARROLLAR EN JAVA?

IDEs (Integrated Development Environments) para desarrollo en Java

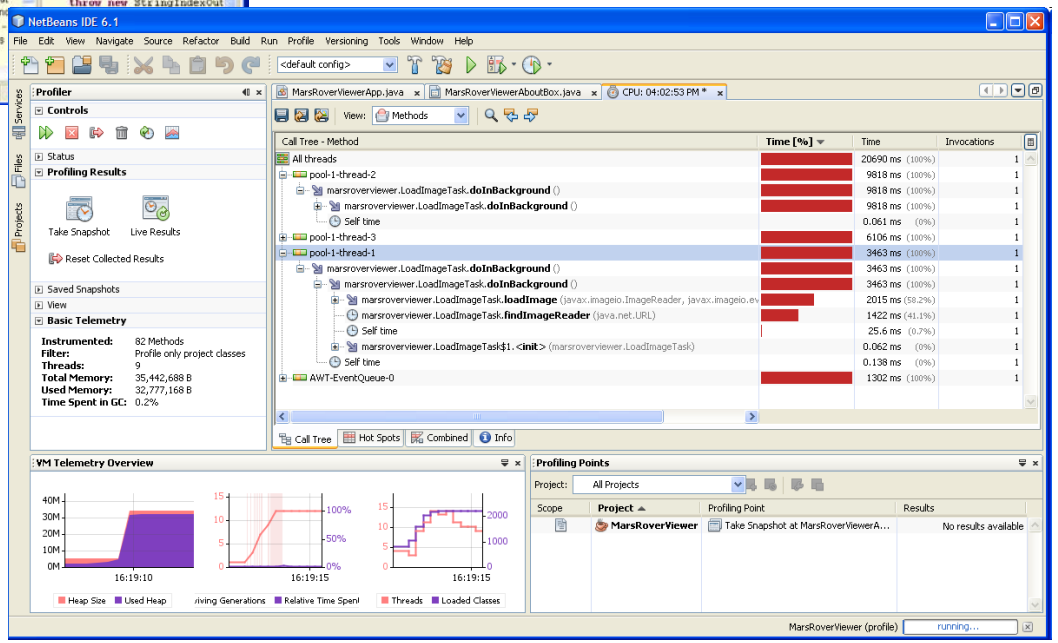


ECLIPSE (.org)

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.

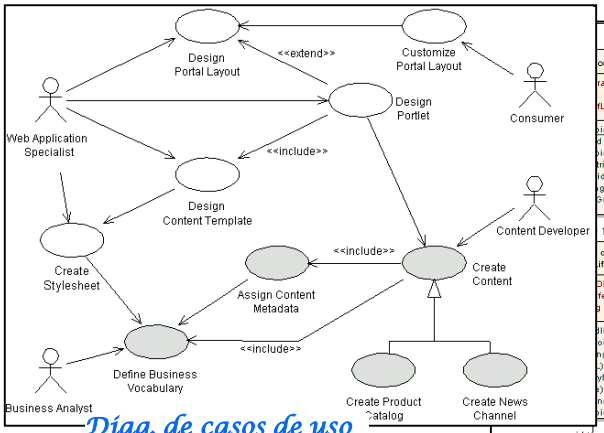


NETBEANS(.org)

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.

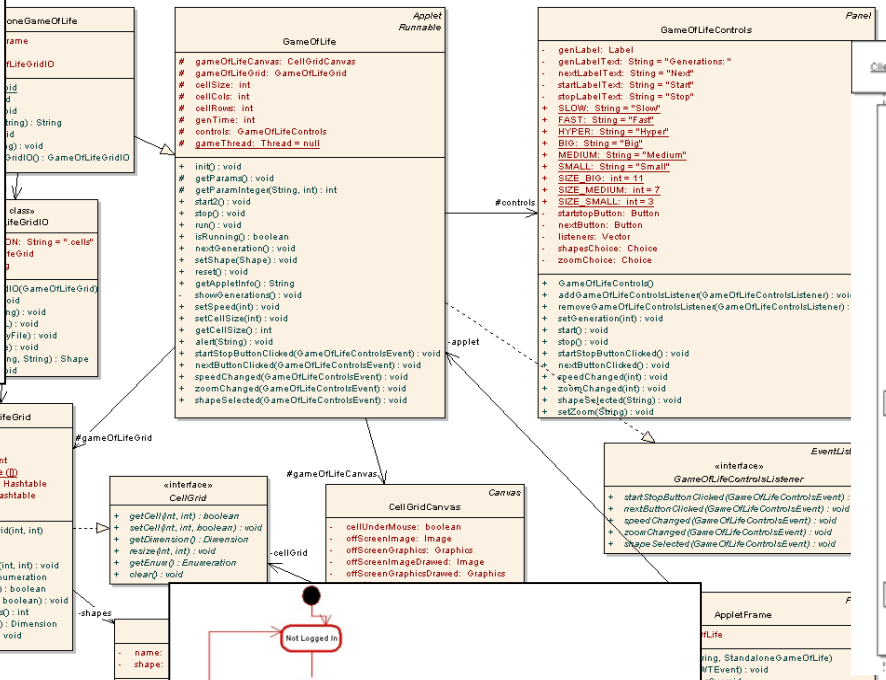
¿Desarrollo en UML?

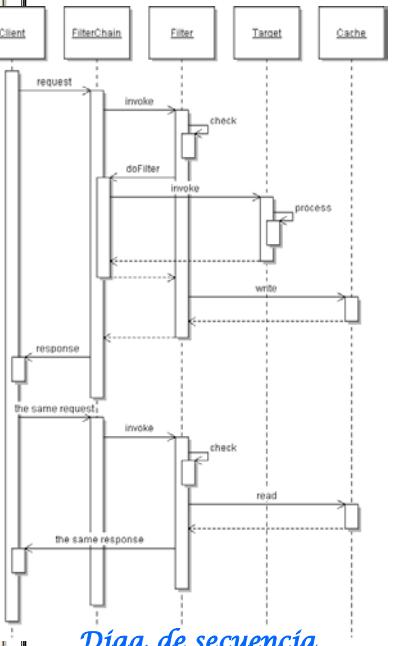
Desarrollo mediante modelado: UML



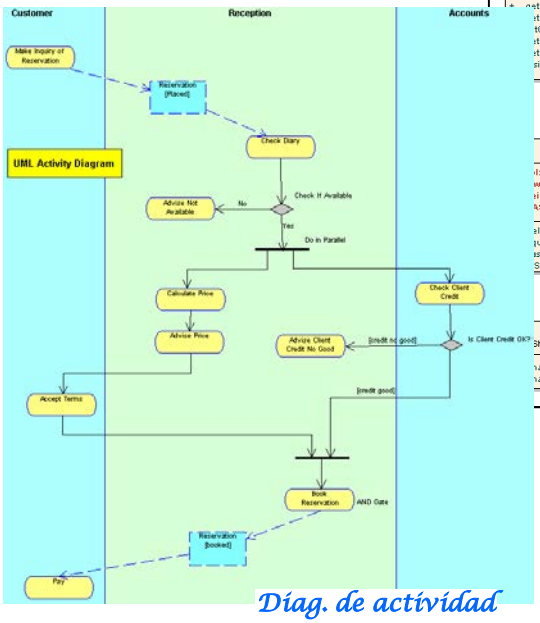
Diag. de casos de uso

Diag. de clases

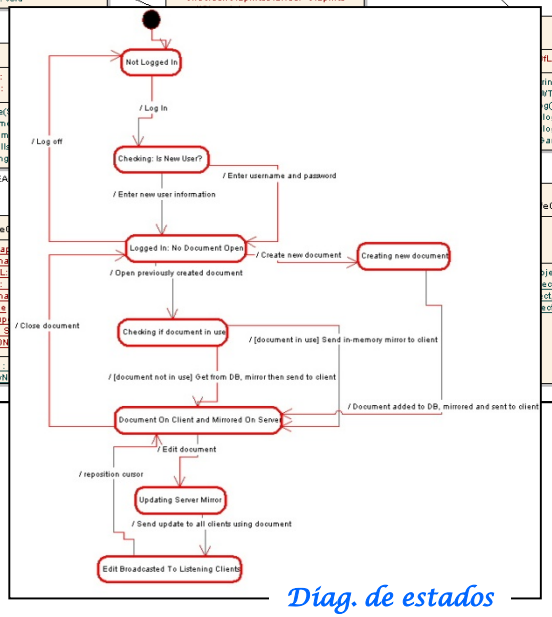




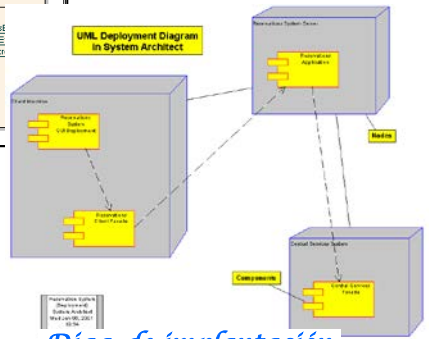
Diag. de secuencia



Diag. de actividad



Diag. de estados



Diag. de implantación