

UT1 Introducción a la programación

Módulo - Programación
Ciclo - Desarrollo de Aplicaciones Web
CI Maria Ana Sanz

Contenidos

- Estructura de un ordenador
- Algoritmos / programas
- Evolución de los lenguajes de programación
- Traductores
 - Compiladores
 - Intérpretes
- Paradigmas de programación
 - procedural
 - orientado a objetos
- Calidad de los programas
- Lenguajes orientados a objetos
- Software orientado a objetos
- UML
- El lenguaje de programación Java
 - Características
 - La plataforma Java
 - Herramientas de desarrollo

Nuestro objetivo

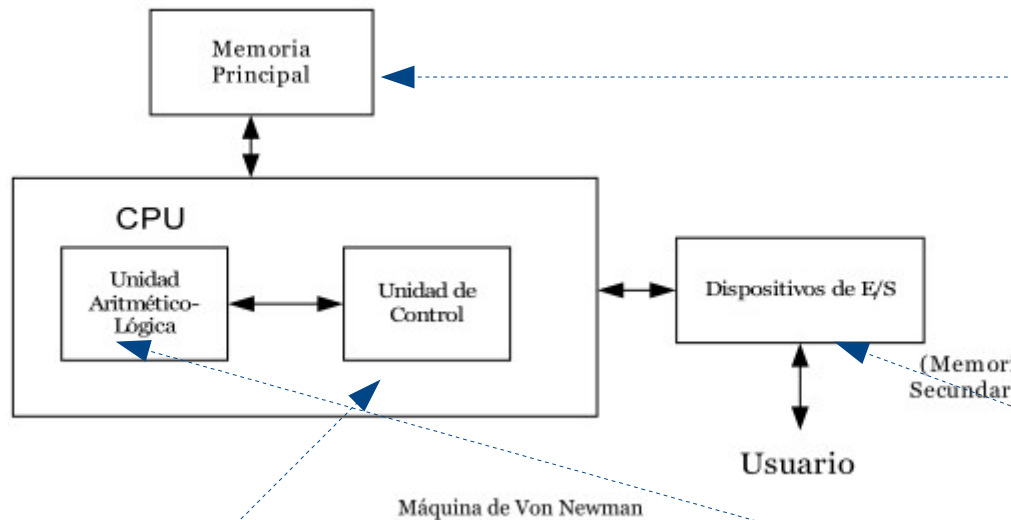
- Aprender a programar
 - ¿Cómo lo haremos?
- Dado un problema
 - diseñar una solución,
 - aplicando el paradigma de la programación orientada a objetos
 - expresando dicha solución en el lenguaje de programación Java
- Paradigma
 - enfoque, pautas para diseñar una solución

Estructura de un ordenador

- Hardware
 - soporte físico
- Software
 - soporte lógico (programas)
 - Sistema operativo
 - Aplicaciones comerciales
 - Programas de usuario

Arquitectura de un ordenador

- Basada en la máquina Von Neumann



Memoria principal – ubicación de los programas. Instrucciones + datos
Volátil. Memoria interna.

Memoria secundaria –
Memoria externa .
Almacenamiento permanente

Dispositivos de E/S –
Interacción con el exterior

Unidad de control – controla la ejecución de las operaciones
Dirige el funcionamiento de los componentes

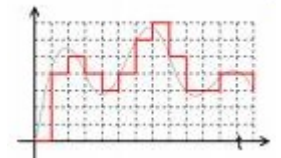
Unidad aritmético-lógica – operaciones aritméticas/lógica y otras más complejas

Unidad de control + Unidad aritmético lógica = CPU (Unidad de proceso)

Arquitectura de un ordenador

- Código binario
 - utilizado por los ordenadores digitales
 - 0 y 1 (bit – binary digit)
- Datos e instrucciones en memoria están en binario
- Lenguaje máquina
 - el único que entiende el ordenador
 - conjunto de instrucciones codificadas en binario

10100010
11110011
00100010
00010010



señal digital binaria

Algoritmos

- **Algoritmo**
 - secuencia ordenada y finita de pasos a seguir para resolver un problema
- Ejemplos de algoritmos
 - pasos para elaborar un receta, pasos a seguir para arrancar un coche,
- Elementos en un algoritmo
 - **procesador** - el que entiende y lleva a cabo el algoritmo (cocinero /ordenador)
 - **entorno** - material necesario para ejecutar el algoritmo (huevos, patatas,... / datos en un programa)
 - **acciones** - actos del procesador sobre el entorno (cascar, freir,... / sumar, restar,)

Algoritmos

- Diagramas de flujo
 - representan algoritmos de forma gráfica

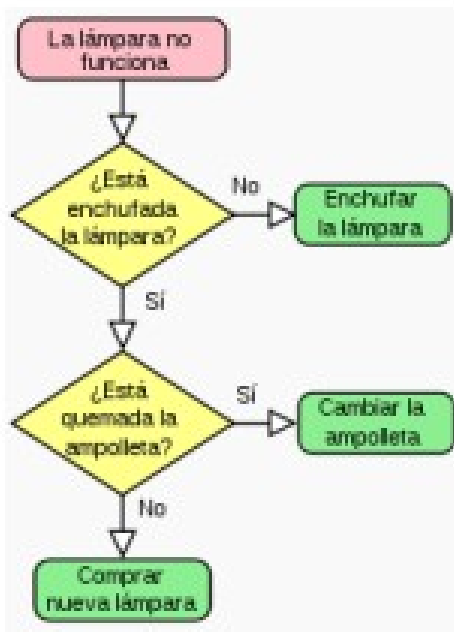


Diagrama de flujo sencillo con los pasos a seguir si una lámpara no funciona.

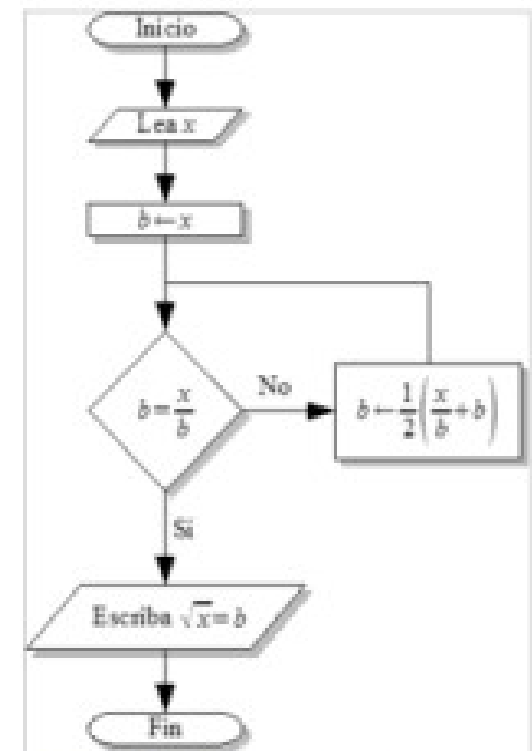


Diagrama de flujo que expresa un algoritmo para calcular la raíz cuadrada de un número x :

Programas

- **Lenguaje de programación**
 - conjunto de símbolos (léxico) y un
 - conjunto de reglas (sintaxis) y una
 - semántica que el ordenador es capaz de entender y procesar
- **Programa**
 - un algoritmo expresado en un lenguaje de programación
 - conjunto de instrucciones que actúan sobre unos datos y que están expresadas en un lenguaje de programación
- **Codificar un programa**
 - expresar un algoritmo en un lenguaje de programación

Clasificación de los lenguajes de programación

- Criterio de *proximidad con la máquina*
 - lenguaje máquina
 - instrucciones codificadas en binario
 - dependiente del hardware
 - difícil de aprender
 - difícil de verificar y poner a punto
 - lenguaje ensamblador
 - versión simbólica del lenguaje máquina
 - cada instrucción se asocia a un símbolo (ADD, SUB, ...)
 - más fácil de programar
 - requiere traducción (con un programa traductor – assembler)
 - dependiente del hardware

Clasificación de los lenguajes de programación

- lenguaje de alto nivel (C, C++, Java, ...)
 - independiente de la máquina (portables)
 - más fácil de aprender
 - más cercano al lenguaje natural
 - más fácil de poner a punto y mantener
 - necesitan ser traducidos a lenguaje máquina – (*compilación ----> compilador*)

Lenguaje de alto nivel	Lenguaje ensamblador	Lenguaje máquina
A = B + C	LDA 0,4,3 LDA 2,3,3 ADD 2,0 STA 0,5,3	021404 031403 143000 041405

Clasificación de los lenguajes de programación

- Criterio *paradigma de programación*
 - imperativos o procedimentales (Pascal, C)
 - orientados a instrucciones
 - orientados a objetos (Java, SmallTalk, C#, Eiffel)
 - centrados en los datos
 - declarativos (funcionales y lógicos)
 - descripciones de funciones matemáticas (Lisp, Haskell) o expresiones lógicas (Prolog)

Traductores – Compiladores e intérpretes

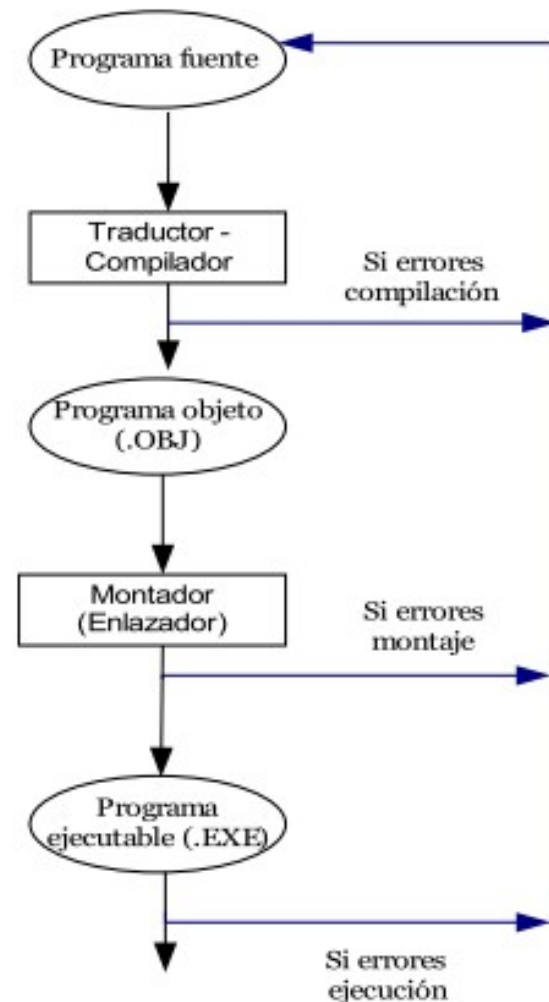
- Programa fuente
 - programa escrito en un lenguaje de programación
- Traductor
 - programa que recibe un programa fuente escrito en un lenguaje de alto nivel y
 - obtiene como salida un programa traducido a código máquina
- Traductores
 - Intérpretes
 - Compiladores

Intérpretes

- Traducen un programa fuente de forma simultánea a su ejecución
 - traducen una instrucción y la ejecutan
 - no guardan el resultado de la traducción (no se genera un ejecutable, **.exe**)
 - errores sintácticos detectados en el momento de la traducción
 - el código fuente es necesario en cada traducción
 - lenguajes interpretados más lentos que los compilados
 - consumen recursos (el intérprete ha de estar en memoria)

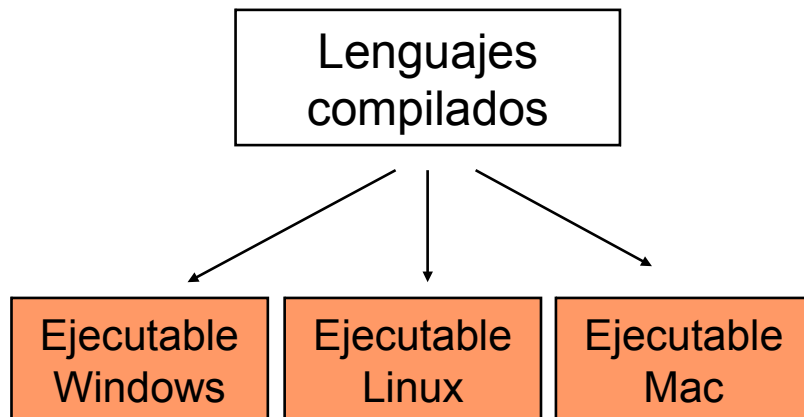
Compiladores

- Traducen un programa fuente a código máquina
 - compilación – el proceso de traducción
 - el traductor se llama *compilador*
 - el compilador detecta errores de sintaxis
 - se guarda el resultado de la traducción (**.obj**)
 - con ayuda del montador (linker) se obtiene un ejecutable (**.exe**) que se guarda



Compiladores

- lenguajes compilados más rápidos de ejecución
- el proceso de traducción solo se realiza una vez
- más dependientes de la plataforma que los lenguajes interpretados



Cada compilador es “dependiente del sistema operativo” (SO): genera un programa ejecutable (.exe) para un SO. Si se quiere ejecutar en otro SO, hay que usar su compilador correspondiente

- Java es un lenguaje compilado e interpretado

Paradigmas de programación

- **Paradigma** – marco de referencia, enfoque de programación para construir programas (pautas, normas, técnicas)
- 4 paradigmas

- funcional

- lógico

- **procedural**

- **orientado a objetos**

utilizan la descomposición (“divide y vencerás”) para afrontar la complejidad de los problemas.

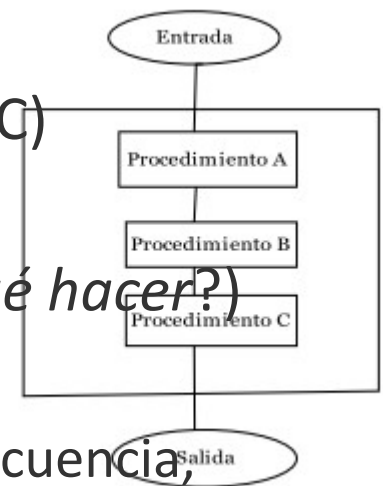
¿Cómo
hacer?

¿Qué
hacer?

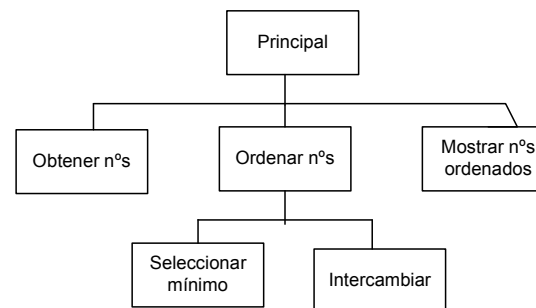
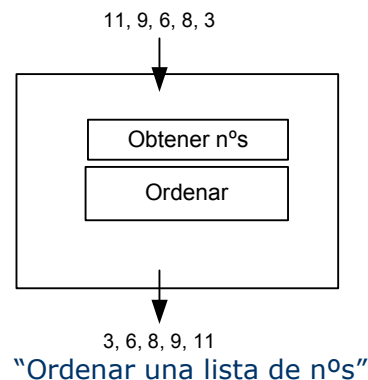
Paradigma procedural (programación estructurada)

■ Procedural

- ampliamente utilizado en la década de los 70 (Pascal, C)
- “*Algoritmos + Estructuras de datos = programas*”
- descomposición funcional o algorítmica (*Cómo hay que hacer?*)
- Bloque de desarrollo – *procedimiento/función*
 - estructuras básicas de la programación estructurada: secuencia, selección e iteración



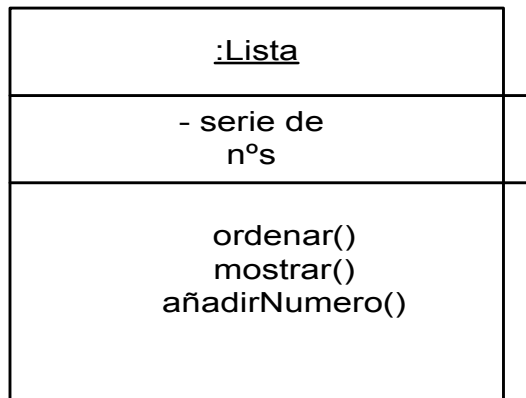
- jerarquía de módulos organizados en torno a uno principal



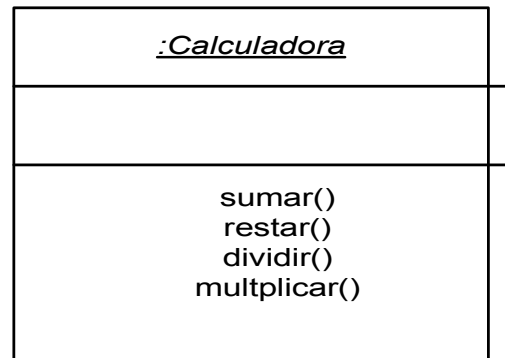
Paradigma orientado a objetos

- Orientado a objetos
 - década de los 90
 - utilizado en todo el desarrollo de software (análisis, diseño, programación)
 - “*Objetos + flujo de mensajes = programas*”
 - para resolver la complejidad de los problemas se utiliza la descomposición en objetos
 - Bloque de construcción del programa – *objeto*
 - un objeto posee identidad, estado y comportamiento
 - un programa es un conjunto de objetos que cooperan entre sí para resolver un problema
 - el objeto es una entidad que se extrae del dominio del problema
 - se identifican objetos y cómo interactuar entre ellos

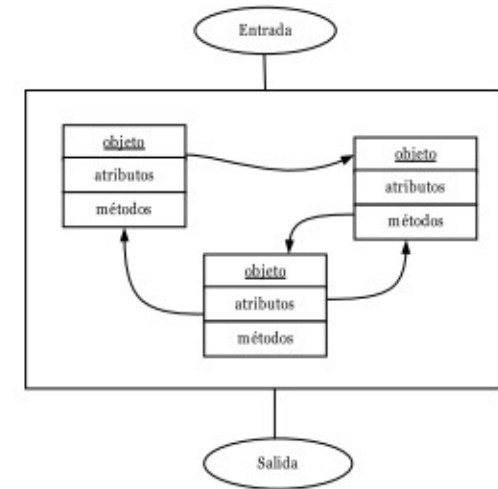
Paradigma orientado a objetos



elemento activo, no pasivo como en la procedural



se piensa de forma distinta, en lugar de pasos a resolver, se piensa e identifican objetos que intervienen en el problema y cómo interactúan



■ Ventajas

- modela mejor el mundo real, más intuitivo
- maneja mejor la complejidad de los problemas
- permite la reutilización (herencia de código)
- programas más robustos y estables (encapsulación), seguridad en los objetos
- facilita la extensibilidad y escalabilidad de los programas

Principios de la P00

■ **Abstracción**

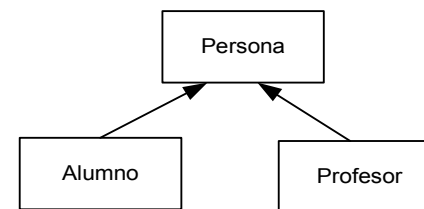
- captar lo esencial ignorando detalles
- manejar la complejidad
- en una aplicación para gestionar un hospital: paciente, médico, especialidad, quirófano

■ **Encapsulación**

- ocultar detalles al exterior
- los objetos son como cajas negras, sabemos qué hacen, no cómo lo hacen. Facilita la reutilización y da robustez al código

■ **Jerarquía**

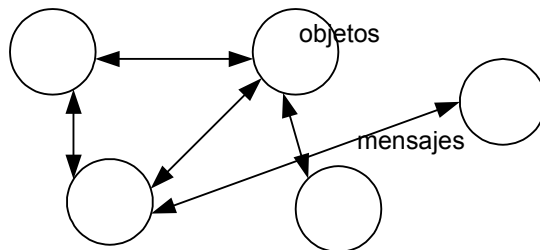
- ordenación de las abstracciones (herencia y polimorfismo)



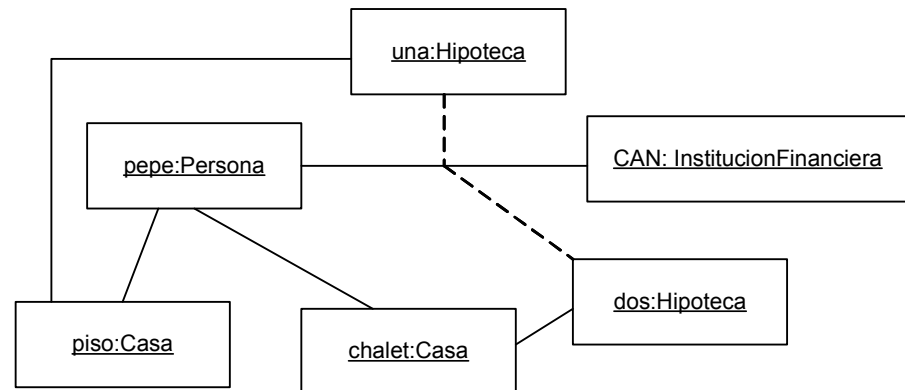
Principios de la P00

■ Modularización

- división del programa en módulos (clases, métodos) para facilitar su diseño, mantenimiento y reutilización



Objetos + flujo de mensajes = Programas

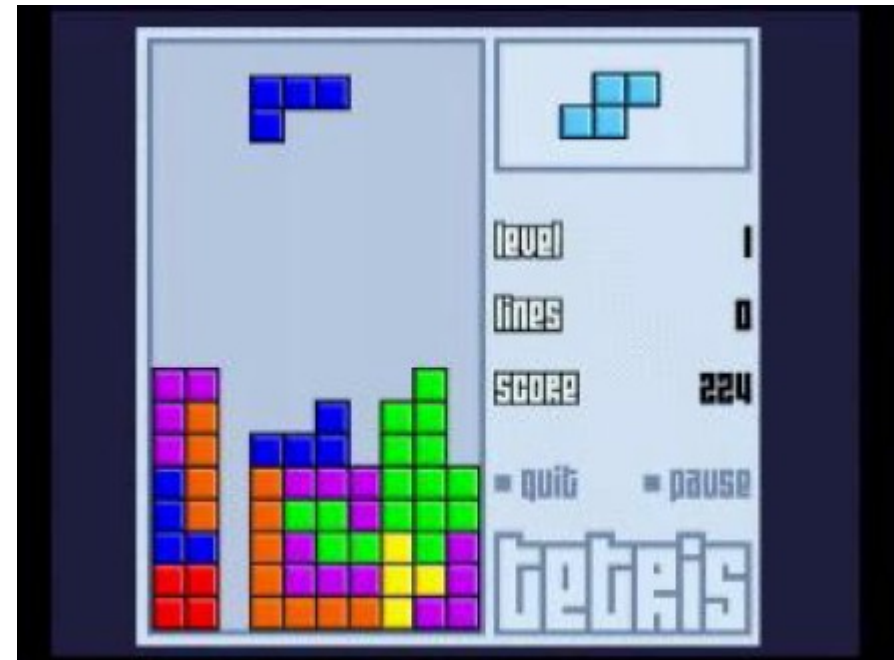


Más sobre la POO

- En la POO:
 - identificar objetos
 - definir estado y comportamiento de los objetos
 - identificar relaciones entre objetos
 - indentificar cómo interactúan los objetos
- Qué puede ser un objeto
 - cosas tangibles – casa, coche, persona
 - cosas intangibles – préstamo, informe, cuenta bancaria, un vuelo

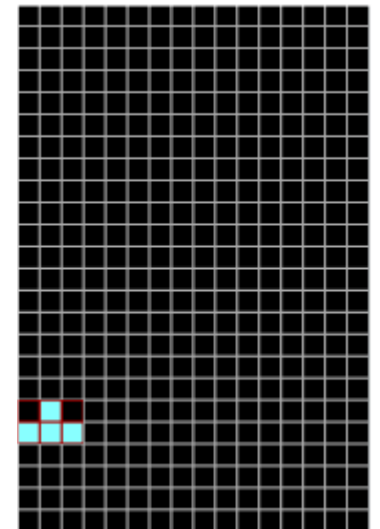
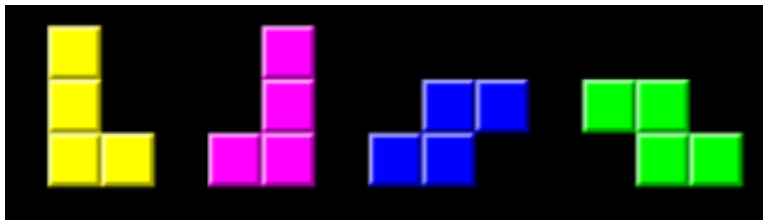
Ejemplo POO - Juegos - Tetris

- Cuáles son los objetos del juego?
- Qué propiedades (estado) tienen esos objetos?
- Qué saben los objetos acerca de lo que deben hacer (comportamiento)?



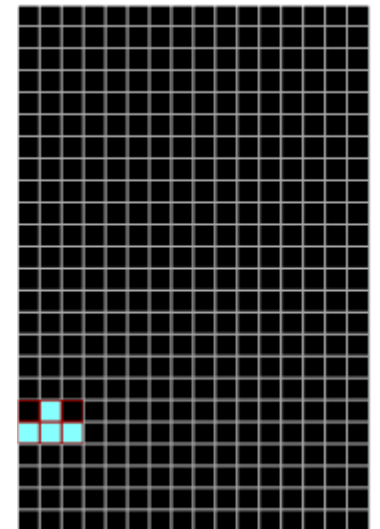
Ejemplo P00 - Juegos - Tetris

- Cuáles son los objetos del juego?
 - pieza, tablero
- Qué propiedades (estado) tienen esos objetos?
 - Pieza
 - orientación, posición, forma, color
 - Tablero
 - tamaño
 - filas



Ejemplo POO - Juegos - Tetris

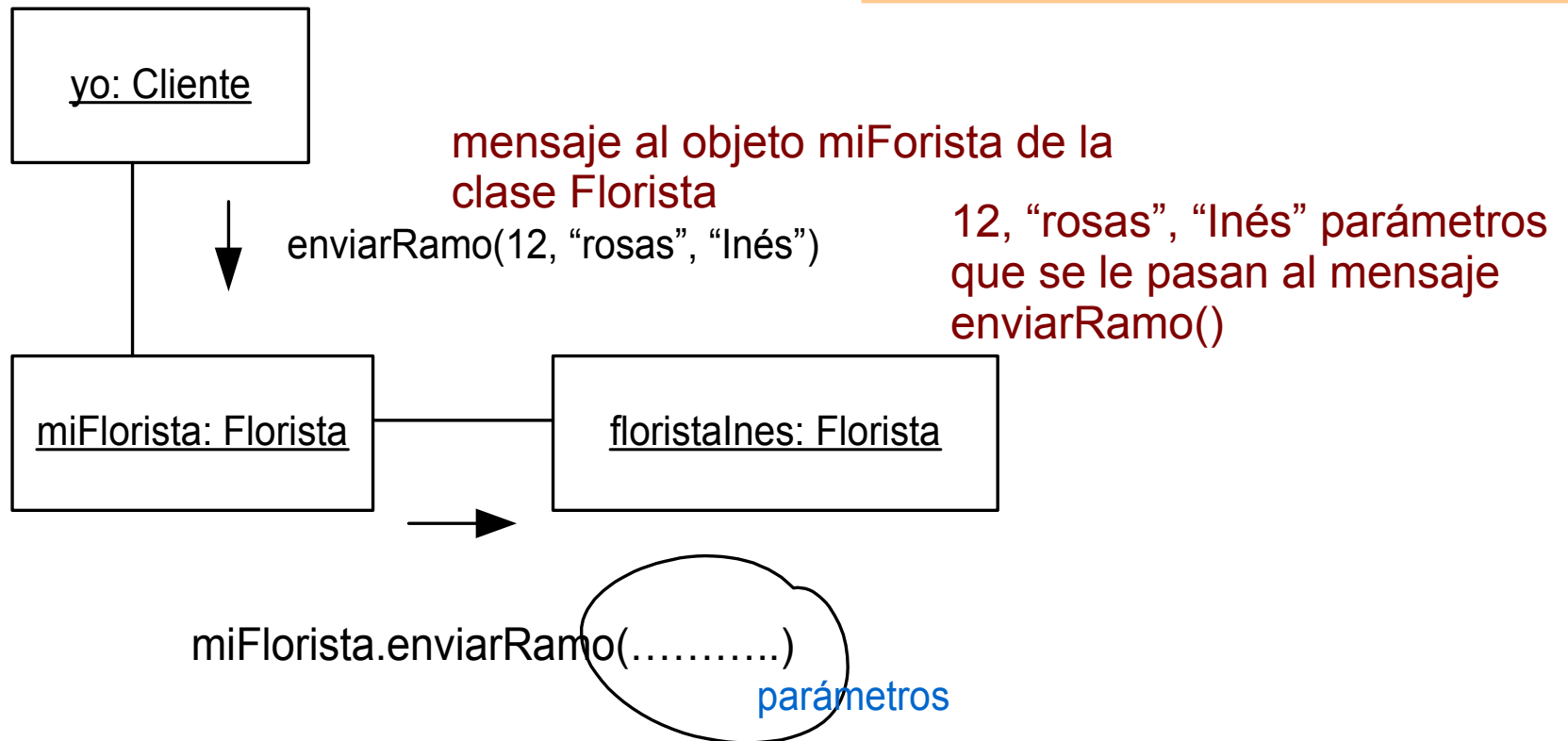
- Qué saben los objetos acerca de lo que deben hacer (comportamiento)?
 - Pieza
 - ser creada
 - caer
 - rotar
 - parar cuando haya colisión
 - Tablero
 - ser creado
 - eliminar filas
 - verificar si fin de juego



Problema de la florista

yo – objeto de la clase Cliente

Leer problema de la florista



Calidad de los programas

- Nuestro objetivo es hacer no solo programas que funcionen (que resuelvan un problema) sino hacer buenos programas
- Características de un buen programa
 - correcto (sin errores, cumpliendo los requisitos)
 - robusto (funciona incluso en situaciones anormales)
 - legible (claro y fácil de leer)
 - modificable (fácil de modificar y mantener)
 - eficiente (utiliza bien los recursos de la máquina, memoria y procesador)
 - reutilizable (fácil de reutilizar, todo o parte)
 - modular (dividido en partes, cada una resolviendo una determinada tarea)

Lenguajes orientados a objetos

- Soportan las características de la POO
 - lenguajes puros
 - proceden de Simula
 - Trabajan exclusivamente con objetos y clases (SmallTalk, Eiffel, Java, C#)
 - lenguajes híbridos
 - basados en lenguajes procedimentales
 - soportan la programación procedural y la POO
 - se construyen a partir de otros ya existentes (Modula y Object Pascal proceden de Pascal, C++ procede de C)

Desarrollo de software orientado a objetos

- Proceso de desarrollo de software
 - conjunto de actividades necesarias para transformar los requerimientos de un problema en el producto software deseado
- Para la construcción de software existen metodologías de desarrollo (RUP – Proceso Unificado de Desarrollo)
 - utiliza el desarrollo iterativo e incremental y UML

Desarrollo de software orientado a objetos

- *Desarrollo iterativo e incremental*
 - dividir un proyecto en miniproyectos
 - cada miniproyecto es una *iteración*
 - en cada iteración se cubre el ciclo entero de desarrollo de una aplicación: *Análisis, diseño, implementación, pruebas y mantenimiento*
 - cada iteración genera una versión parcialmente completa de un sistema
 - sucesivas iteraciones se construyen unas sobre otras hasta completar el sistema
 - incremento: diferencia entre una y otra iteración
- Ejemplo – “*diseñar un reproductor MP5*”
 - 1ª iteración - “reproducir una canción”
 - análisis, diseño, implementación,
 - 2ª iteración - “seleccionar aleatoriamente una canción”
 - análisis, diseño, implementación,

Ciclo de desarrollo de software



- Análisis OO – investigar el problema y requisitos
 - *“¿cómo es la casa de mis sueños?”* **QUÉ**
 - diagrama de clases del dominio
- Diseño OO – buscar solución que satisfaga los requisitos
 - *“¿cómo voy a construir la casa?”* - **CÓMO**
 - diagrama de clases software
- Implementación – Programación OO – codificación *
 - *“construyo la casa”*
 - codifico el diseño en un lenguaje

Ciclo de desarrollo de software

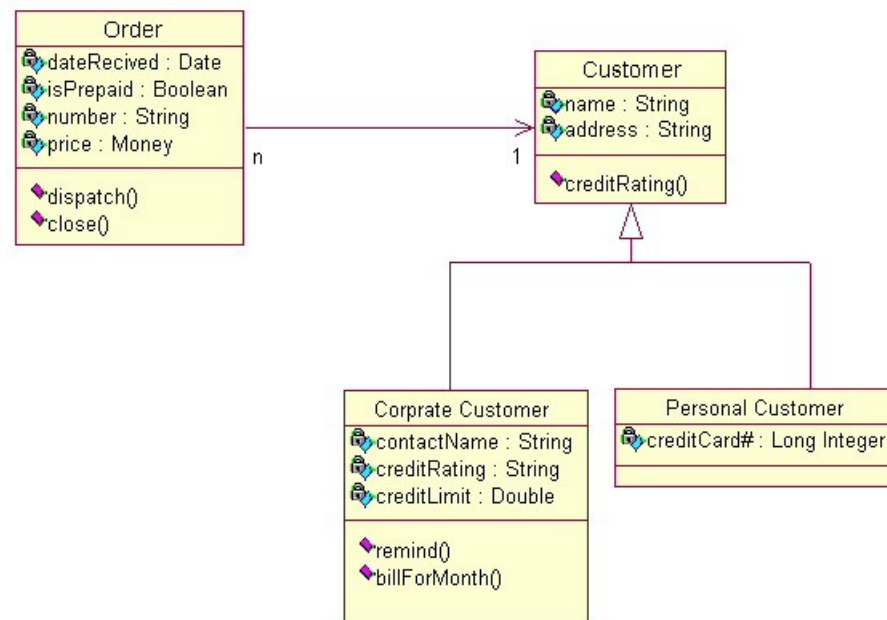
- Pruebas
 - el producto construido hace lo especificado
 - se hacen tests con datos de prueba para comprobar el funcionamiento correcto
- Mantenimiento
 - modificación y adaptación del producto debido a errores o a nuevas necesidades

UML

- UML (*Unified Modeling Language* – Lenguaje unificado de modelado)
 - lenguaje para describir modelos
 - modelo – descripción abstracta de un sistema, representación simplificada del mismo
 - no es una metodología de desarrollo de software
 - notación para especificar, construir, visualizar y documentar los elementos de un sistema de software

UML

- UML describe una serie de diagramas o representaciones gráficas
 - diagrama de clases (estructura estática de un modelo, las clases y sus relaciones)



- Extensión Bluej para UML (UML Extension – muy básica)

El lenguaje de programación Java. Orígenes.

- **Java** – lenguaje OO de alto nivel
 - desarrollado por el equipo de James Gosling de Sun Microsystems en 1995 (ahora Sun ha sido adquirida por Oracle)
 - diseñado inicialmente para pequeños dispositivos electrodomésticos (microondas, calculadoras y TV interactiva)
 - se llamaba inicialmente *Oak* (roble)
 - los proyectos no tuvieron éxito y Java quedó en el olvido
 - www – con el boom de Internet se dieron cuenta de que Java era idóneo para desarrollar aplicaciones web (neutralidad de la plataforma, ejecución de *applets* en el navegador)
 - *Duke* – mascota de Java

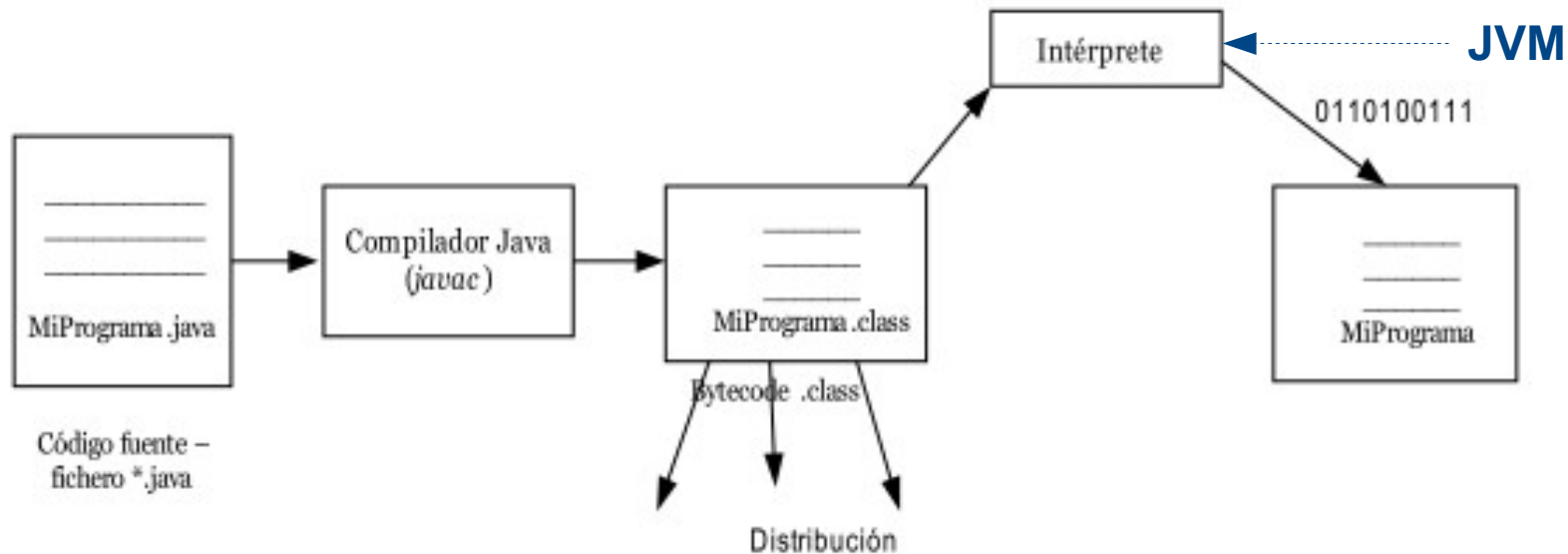


Java. Características.

- Sencillo
 - sintaxis sencilla, pocas construcciones del programa, sin punteros, sin herencia múltiple
- Orientado a objetos
- Distribuido
 - arquitectura inherente al trabajo en entornos de red
- Robusto – confiable
 - gestiona excepciones en tiempo de ejecución
 - verificación temprana de errores
- Seguro
 - mecanismos de seguridad para proteger el sistema

Java. Características.

- Interpretado – no es un lenguaje compilado al uso (como C)
 - Java es compilado e interpretado a la vez
 - necesitamos un intérprete para ejecutar los programas Java

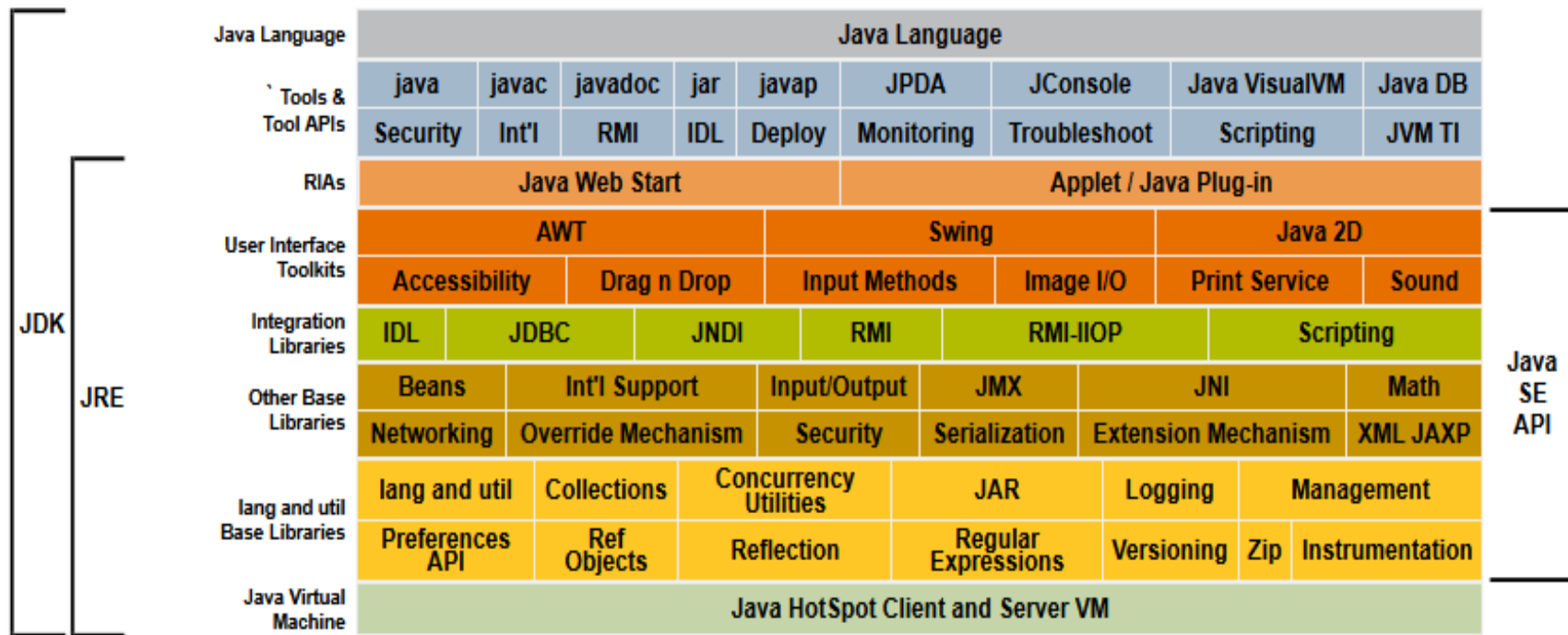


ficheros *.class - no contienen código máquina comprensible por ningún procesador sino bytecodes, código de bajo nivel (un lenguaje intermedio) que será interpretado por la máquina virtual de Java. Esta máquina abstracta (JVM) es la que ha de estar instalada en una máquina real para poder ejecutar la aplicación java. El código de bytecodes es el mismo para todos los sistemas.

Java. Características.

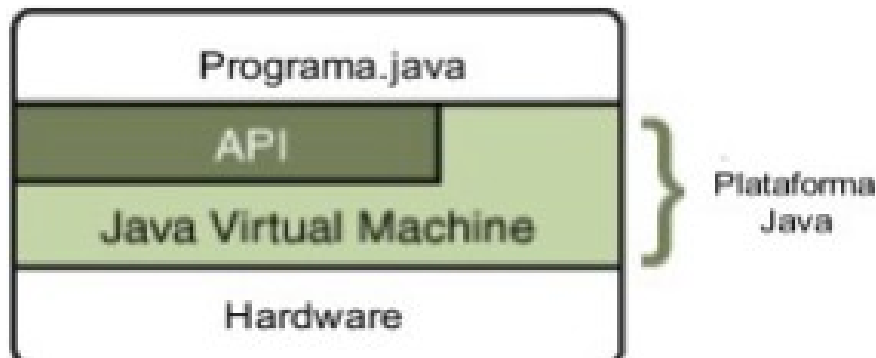
- Arquitectura neutral y portable
 - independiente de la plataforma en que se ejecuta
 - *“escribe una vez, ejecuta en cualquier lugar”*
- Multihilo
 - ejecución de varias tareas simultáneamente

La plataforma Java.



<http://download.oracle.com/javase/8/docs/>

La plataforma Java.

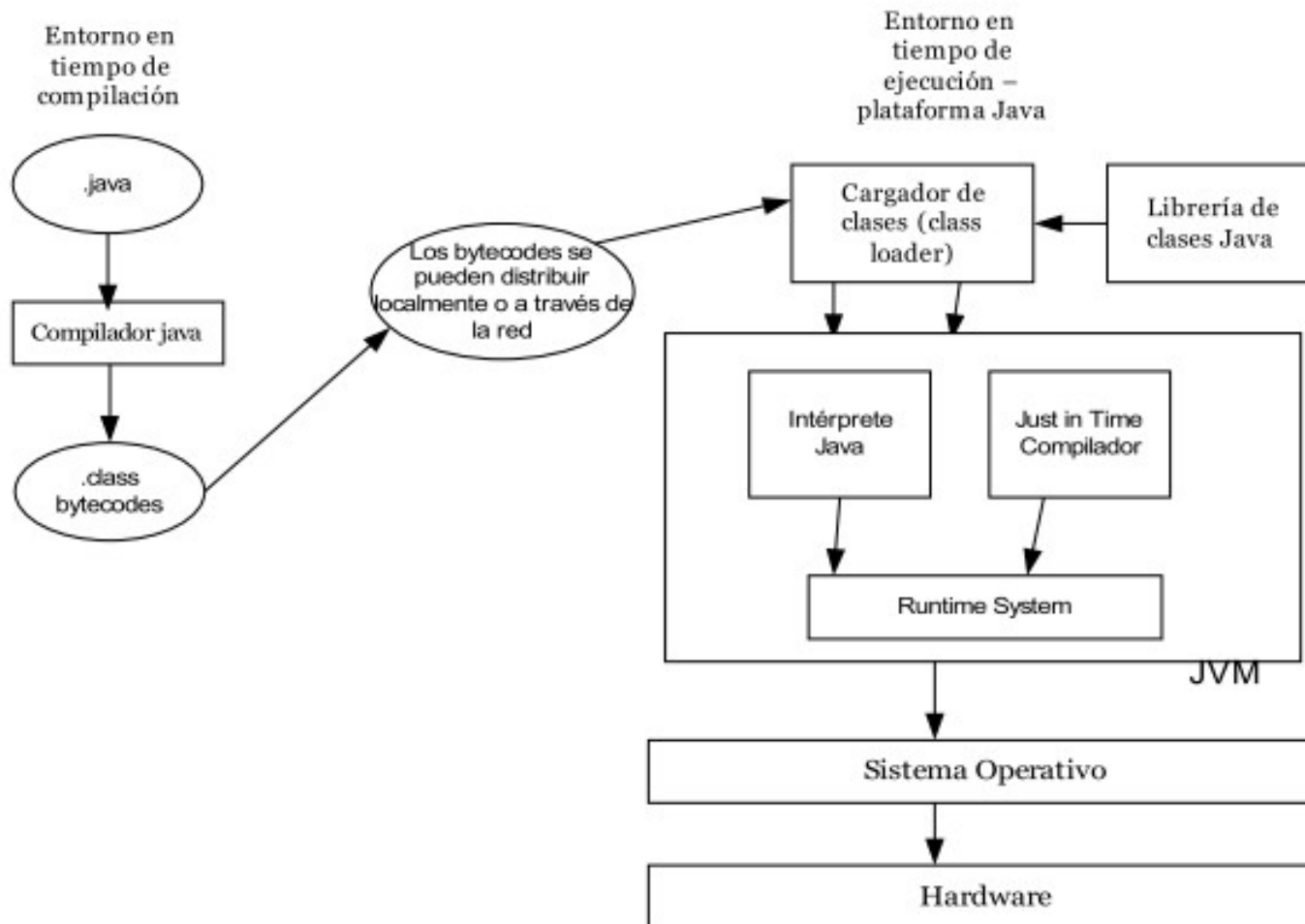


- Java no solo es un lenguaje sino toda una plataforma de desarrollo
 1. JVM - Máquina virtual (*Java virtual machine*)
 2. API de Java – (*Java Application Programming Interface*)
- **JVM + API = JRE (Java Runtime Environment)**

- JVM

- software
- lee ficheros .class de bytecodes y los traduce a código máquina
- necesaria para ejecutar un programa Java
- dependiente de la plataforma (Linux, Windows, ...)
- la contienen también los navegadores

La plataforma Java.



La plataforma Java.

- API de Java
 - colección de componentes software, clases, que el programador puede incluir en sus programas.
 - se agrupan en librerías de clases relacionadas (*paquetes*).
- Para ejecutar programas Java simplemente (no para desarrollarlos) es suficiente tener instalado el JRE en el sistema.

Herramientas de desarrollo. Java SDK (Java Software Development Kit)

- Para desarrollar programas Java
 - **javac** – compilador a bytecodes
 - **javadoc** – generador de documentación
 - **java** – llamada a la JVM (lanzador de aplicaciones Java)
 - **jar** – crear archivos .jar
 - **jdb** – depurador de consola
- Estas herramientas junto con el JRE (además del propio lenguaje Java) constituyen el Java SE Development Kit (JDK).
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Versiones de Java

Java 1.0

Java 1.1

Cambios significativos

J2SE 1.2

J2SE 1.3

J2SE 1.4

J2SE 5.0

Conocida como Java 5.0 (Tiger). Cambio cualitativo (colecciones genéricas, for-each, enumerados, autoboxing/unboxing, ...)

Java SE 6

Diciembre 2006. Primera versión para trabajar con Windows Vista. Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Importantes mejoras (mejoras en la interfaz gráfica y en el rendimiento, incluye un cliente completo de Servicios Web,)

Java SE 7

Julio 2011. Dolphin. Primera versión lanzada desde que Oracle adquiriese Sun. API de New File System, que ofrece acceso nativo a varias operaciones sobre archivos y carpetas, mejoras en seguridad y en conectividad, además de incluir mejoras en el soporte para internacionalización,

Versiones de Java

Java SE 8

Marzo 2014. Notables cambios en el lenguaje y mejoras en la plataforma (introducción de las expresiones *lambda* (o *closures*) que da a Java la característica de programación funcional de otros lenguajes como Phyton, nueva API para fechas, mejoras de seguridad,)

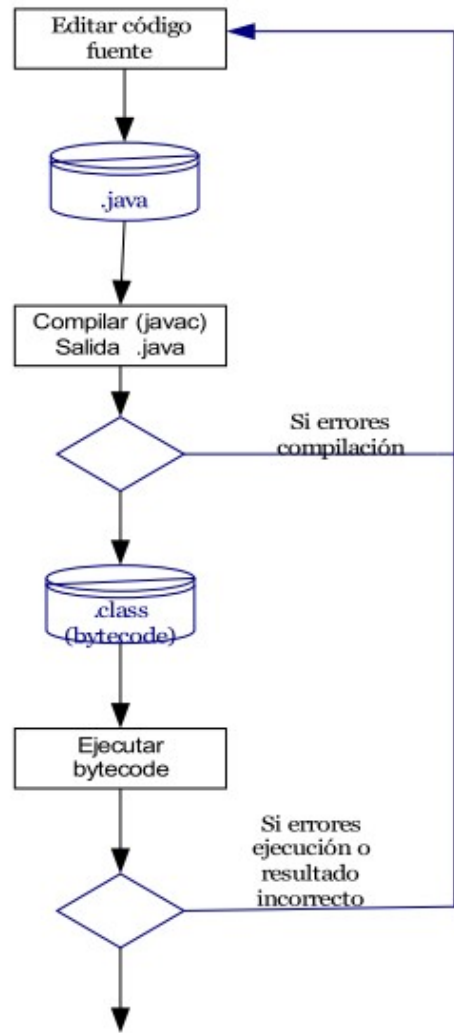
Ediciones Java

- Diferentes ediciones de Java para diferentes tipos de aplicaciones
 - **Java SE** - escribir, desarrollar y ejecutar applets, aplicaciones de escritorio y aplicaciones web Java
 - **Java EE** – versión ampliada de Java SE . Incluye API para aplicaciones con arquitectura multicapa, aplicaciones distribuidas, servicios web.
 - **Java ME** – edición para dispositivos móviles

Tipos de aplicaciones Java

- **Aplicaciones normales**
 - aplicaciones autónomas (*stand-alone*)
 - salida texto o gráfica
- **Applets**
 - se ejecutan dentro de un navegador
 - salida gráfica
- **Aplicaciones web**
 - se ejecutan en un servidor
 - tecnologías de *servlets* / JSP
 - interactúan normalmente con una base de datos

Editar - compilar - ejecutar un programa Java



- Editar el código fuente (con un editor de textos). **.java**
- Llamar al compilador **javac** para obtener **.class** (fichero de bytecodes)
- Llamar al intérprete (la JVM), **java**, para traducir y ejecutar los bytecodes

Entornos de desarrollo

- Facilitan las tareas de edición, compilación, depuración, ... de los programas
- **BlueJ** – el que utilizaremos la mayor parte del curso
 - entorno diseñado para la enseñanza y aprendizaje de la POO en Java
 - gratuito
 - desarrollado por las universidades de Kent, Deakin y Southern
 - no es un entorno de desarrollo profesional
 - <http://www.bluej.org/>



Entornos de desarrollo

- **NetBeans**

- entorno de desarrollo integrado profesional
- gratuito
- plugin para trabajar con proyectos BlueJ
- <http://netbeans.org/>



- **Eclipse** –

- entorno de desarrollo integrado de código abierto multiplataforma
- emplea módulos (plugin) para proporcionar toda su funcionalidad.
- <http://www.eclipse.org/>



Enlaces interesantes

- <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>
 - El ranking TIOBE es un instrumento de referencia que indica la popularidad de los lenguajes de programación
- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
 - Página oficial de Java
- <http://www.bluej.org/>
 - Página oficial de BlueJ
- <http://netbeans.org/>
 - Página oficial de NetBeans
- <http://www.eclipse.org/>
 - Página oficial de Eclipse

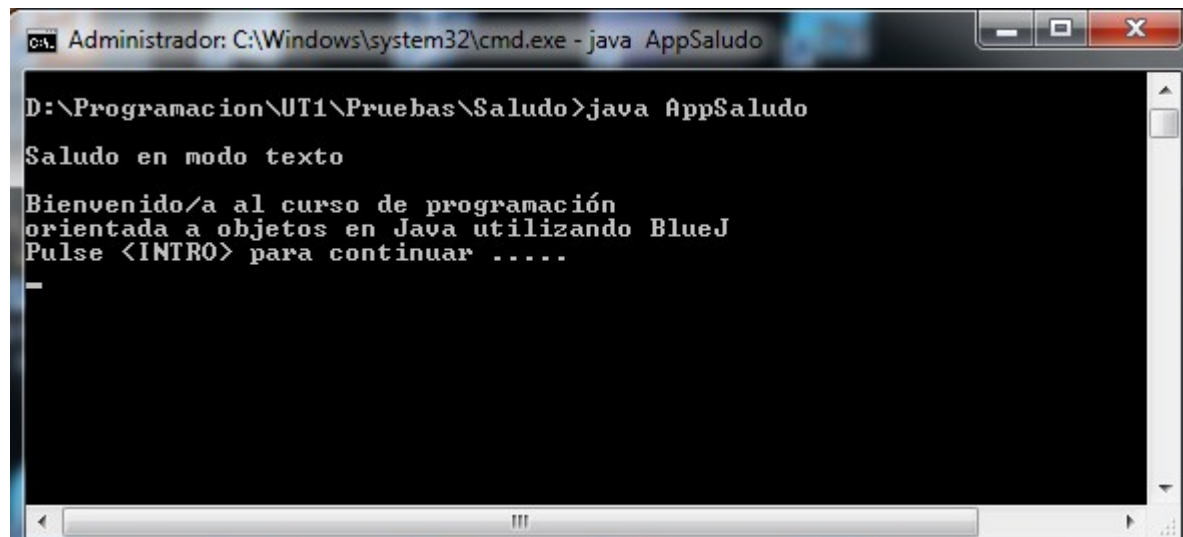
Instalación

Probando desde línea de comandos

- Instalación y configuración de Java SE Development kit
- Proyecto Saludo (dentro de la carpeta Pruebas)
 - (ver Práctica I en Moodle)
 - Inicio / Ejecutar / cmd
 - Nos cambiamos al directorio
 - `cd C:\Programacion\Ut1\pruebas\saludo`
 - `C:\Programacion\Ut1\pruebas\saludo> javac AppSaludo.java`
 - `C:\Programacion\Ut1\pruebas\saludo> java AppSaludo`

Instalación

Probando desde línea de comandos



A screenshot of a Windows command prompt window. The title bar reads "Administrador: C:\Windows\system32\cmd.exe - java AppSaludo". The command prompt shows the following text:

```
D:\Programacion\UT1\Pruebas\Saludo>java AppSaludo
Saludo en modo texto
Bienvenido/a al curso de programación
orientada a objetos en Java utilizando BlueJ
Pulse <INTRO> para continuar .....
```

The cursor is positioned on the line following the prompt.

