Programación Orientada a Objetos

Tema 1:
Introducción a los lenguajes de programación

Contenidos

- Tema 1: Introducción a los lenguajes de programación

1. Historia de los lenguajes de programación
2. Paradigmas de programación
3. Intérpretes y compiladores
4. Fases de la traducción de lenguajes
5. Máquinas virtuales
6. Anexo: Introducción al lenguaje de programación Java

HISTORIA DE LOS LENGUAJES DE PROGRAMACIÓN (I)

- Cualquier programa ha de escribirse en un lenguaje entendible por el ordenador
- El ***ordenador:***
  - Máquina electrónica (no habla nuestro idioma).
  - Dotada principalmente de memoria y un procesador, que es el encargado de interpretar el algoritmo, ejecutando las operaciones correspondientes a cada paso.
    - El proceso de elaboración de un programa o software se denomina ***programación.***
    - El conjunto de instrucciones que se pueden utilizar para construir un programa se denomina ***lenguaje de programación.***
- Representación de los datos
  - Se realiza de forma binaria (BIT: 0 ó 1).
  - Esta representación está basada en la capacidad que tiene el ordenador de diferenciar dos estados o niveles de voltaje.

HISTORIA DE LOS LENGUAJES DE PROGRAMACIÓN (II)

- ***Lenguajes máquina:***
  - El único que entiende directamente el ordenador.
  - Están compuestos por cadenas binarias (cadenas de 0s y 1s) que especifican una serie de operaciones y las posiciones de memoria implicadas en la operación.
  - El código máquina es conocido como código binario, aunque se emplea el sistema hexadecimal para simplificar el trabajo de escritura.
  - ***Inconvenientes***
    - Dificultad y lentitud en la codificación.
    - Poca fiabilidad.
    - Gran dificultad para verificar y poner a punto los programas.
    - Los programas solo son ejecutables en el mismo procesador.
  - La única ***ventaja*** de programar en lenguaje máquina
    - Posibilidad de cargar un programa en memoria sin necesidad de traducción posterior ya que el código es perfectamente inteligible por el computador.
  - Escribir un programa utilizando cadenas de 0 y 1 implicaría un tiempo elevado y una gran probabilidad de cometer errores debido a la dificultad y lentitud en la codificación. Además los programas sólo son ejecutables en el mismo procesador para el que se ha realizado la codificación. Por esta razón, se desarrollaron los ***lenguajes de ensamblador.***

HISTORIA DE LOS LENGUAJES DE PROGRAMACIÓN (III)

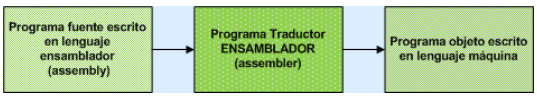
- ***Lenguajes ensambladores o de bajo nivel***

  – Indican al ordenador las operaciones que ha de realizar mediante la utilización de códigos nemotécnicos.
  – Son más fáciles de utilizar que los lenguajes máquina, pero, al igual que ellos, dependen de la máquina en cuestión.
  – Un programa realizado en lenguaje ensamblador no puede ser ejecutado directamente por la computadora, sino que necesita ser traducido a lenguaje máquina (código binario).
  – El programa utilizado para realizar esta conversión se denomina programa ensamblador y se encarga de obtener el programa objeto en código máquina a partir del programa fuente en ensamblador.

  – Inconvenientes: Dependencia total de la máquina, lo que impide la portabilidad de la aplicación entre máquinas con procesadores diferentes. Su utilización requiere conocer bien el hardware.

  – Ventajas :
    - Respecto al código máquina: mayor facilidad de codificación/modificación, ahorran tiempo y requieren menos atención a detalles.
    - Respecto a lenguajes de alto nivel: velocidad de ejecución.

  – Aplicaciones reducidas: Tiempo real, control de procesos y de dispositivos electrónicos.



HISTORIA DE LOS LENGUAJES DE PROGRAMACIÓN (IV)

- ***Lenguaje de alto nivel***

  – Hoy en día son los más utilizados. Están orientados al problema.
  – Permiten una mejor comprensión de los programas debido al empleo de una terminología que se aproxima al lenguaje humano.
  – Los programas son independientes de la máquina, lo que nos permitirá ejecutarlos en diferentes máquinas sin ninguna o pocas modificaciones.
  – El ordenador no es capaz de reconocer directamente estas ordenes, por lo que es necesaria su traducción a un lenguaje que el sistema pueda entender.

  – ***Ventajas***

    - Tiempo de formación de los programadores es relativamente corto comparado con otros lenguajes.
    - La escritura de programas se basa en reglas sintácticas similares a los lenguajes humanos.
    - Las modificaciones y puestas a punto de los programas son más fáciles.
    - Reducción del costo de los programas.
    - Transportabilidad.
    - Permiten tener una mejor documentación.
    - Fáciles de mantener.

  – ***Desventajas***

    - No se aprovechan los recursos internos de la maquina que se explotan mucho mejor en lenguajes máquina y ensambladores.
    - Aumento de la ocupación de memoria.
    - Tiempo de ejecución de los programas es mucho mayor.

PARADIGMAS DE PROGRAMACIÓN

- Los paradigmas o modelos de programación son un conjunto de métodos sistemáticos que se aplican en todos los niveles del diseño de software. Atienden a enfoques específicos para la construcción de programas.
- Cada paradigma de programación tiene sus ventajas e inconvenientes, y la elección correcta del paradigma a utilizar depende del tipo de aplicación que se persiga.
- Un lenguaje de programación puede soportar uno o varios paradigmas.
- Los paradigmas más frecuentemente utilizados son:
  - Imperativo.
  - Funcional.
  - Lógico.
  - Orientado a objetos.

PROGRAMACIÓN IMPERATIVA

- Este paradigma utiliza una serie de instrucciones o sentencias que definen cambios en el estado de un programa.
- El programador codifica algoritmos (secuencias de pasos bien definidas en las que se especifican las operaciones a realizar durante el transcurso de la ejecución).
- Puede existir tanto a bajo nivel (código máquina) como a alto nivel.
- Ejemplos de lenguajes que soportan el paradigma imperativo:
  - Fortran
  - Pascal
  - C
  - Cobol

PROGRAMACIÓN FUNCIONAL

- Este paradigma utiliza la definición de funciones matemáticas y su aplicación.
- No existen asignaciones de variables o construcciones estructuradas como la secuencia o la iteración (aunque en la práctica algunos lenguajes las adoptan, dando lugar a soluciones híbridas).
- El programador codifica funciones y reglas.
- En el paradigma funcional puro, la repetición se lleva a cabo mediante el uso de la recursión.
- Ejemplos de lenguajes que soportan el paradigma funcional:
  - Lisp
  - Haskell
  - Caml
  - Miranda

PROGRAMACIÓN LÓGICA

- Este paradigma utiliza la aplicación de la lógica matemática para la representación de los datos y la inferencia de resultados aplicando a esos datos una serie de reglas que los relacionan.
- El programador codifica predicados y reglas.
- Es utilizado en el diseño de sistemas expertos, demostración de teoremas, aplicaciones relacionadas con el lenguaje natural, etc.
- Enfoque orientado a la descripción del problema, no a cómo resolverlo.
- Ejemplos de lenguajes que soportan el paradigma de programación lógica:
  - Prolog

PROGRAMACIÓN ORIENTADA A OBJETOS

- Este paradigma se basa en la utilización de entidades denominadas objetos y de las interacciones entre objetos para el diseño de una aplicación.
- Se busca otorgar una gran independencia a las distintas partes que componen una aplicación.
- Cada objeto definido en la aplicación posee una identidad (se diferencia del resto), y contiene datos y algoritmos propios que determinan las operaciones que puede realizar.
- Ejemplos de lenguajes que soportan el paradigma de programación orientada a objetos:
  - Java
  - C++
  - C#
  - Eiffel
  - Smalltalk

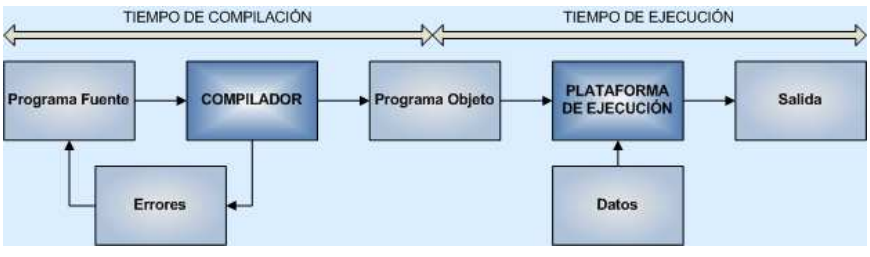
INTÉRPRETES Y COMPILADORES (I)

- Los programas fuente escritos en lenguajes de alto nivel necesitan ser traducidos a código máquina para su ejecución. Esta traducción se lleva a cabo por los programas traductores de lenguaje, que se dividen en **intérpretes** y **compiladores**.
- Un intérprete es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta. Además no se genera un programa escrito en lenguaje máquina y, a medida que se van introduciendo líneas de código, éstas son verificadas.
- Un compilador es un programa que traduce los programas fuentes escritos en lenguaje de alto nivel a lenguaje máquina.
- Típicamente, la velocidad de ejecución de programas compilados es superior a la de los programas interpretados.

INTÉRPRETES Y COMPILADORES (II)

FASES DE LA TRADUCCIÓN DE LENGUAJES

- Proceso por el cual se traducen las instrucciones escritas en un determinado lenguaje de programación a lenguaje máquina, el cual es interpretado por la computadora.
- Los traductores transforman un texto escrito en un lenguaje (fuente) a otro texto en un lenguaje distinto (objeto), manteniendo el significado del texto original.
- Es posible distinguir en el proceso de elaboración de un programa, desde su escritura hasta su ejecución en una plataforma, dos periodos de tiempo:
  - El tiempo de compilación
  - El tiempo de ejecución



FASES DEL PROCESO DE COMPILACIÓN (I)

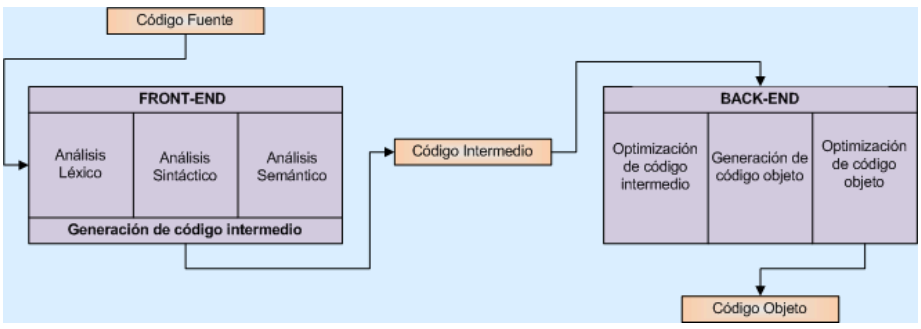
– ***Fase de análisis*** (front-end)

- Depende del lenguaje fuente, es independiente del lenguaje objeto.
- En esta fase el programa es descompuesto en sus elementos fundamentales:
  - Análisis léxico.
  - Análisis sintáctico.
  - Análisis semántico.
  - Verifica si el programa en lenguaje fuente es correcto.
- Los errores detectados son notificados a través del gestor de errores.

FASES DEL PROCESO DE COMPILACIÓN (II)

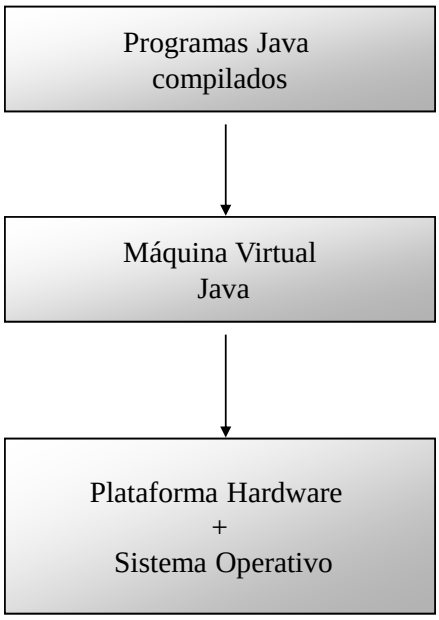
– ***Fase de síntesis*** (back-end)

- Depende del lenguaje objeto.
- Es independiente del lenguaje fuente.
- Esta fase se lleva a cabo después de que el modulo de análisis ha verificado que el código fuente es correcto.
- Generación del código objeto.



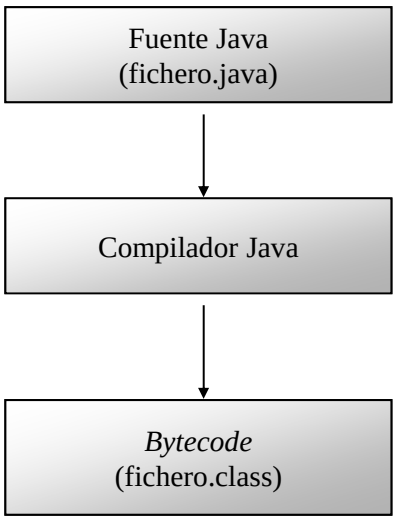
MÁQUINAS VIRTUALES (I)

- Una máquina virtual es una implementación software que permite la ejecución de programas sobre plataformas (hardware, sistema operativo, etc.) que pueden ser distintas a la utilizada por el usuario.
- Pueden proporcionar una arquitectura de instrucciones propia para simular un hardware concreto (MV de sistema), o bien ejecutarse como un proceso más dentro de un sistema operativo (MV de proceso) y ejecutar un único proceso, cerrándose cuando el proceso finaliza.
- En el caso de Java, cualquier programa escrito en este lenguaje y compilado podrá ser ejecutado en cualquier ordenador, independientemente de la arquitectura de éste, siempre que se tenga instalada la correspondiente máquina virtual.



MÁQUINAS VIRTUALES (II)

- Las aplicaciones escritas en Java son compiladas y convertidas a *bytecode*, que consiste en un código intermedio más abstracto que el código máquina para proporcionar mayor independencia de la plataforma.
- Cada vez que se ejecuta la aplicación, la máquina virtual de Java interpreta el bytecode previamente generado.
- Se han desarrollado varias implementaciones alternativas (para distintos sistemas operativos).
- Existen compiladores just-in-time que traducen el bytecode de Java a código máquina justo antes de la ejecución, con vistas a aumentar la velocidad.



Anexo:

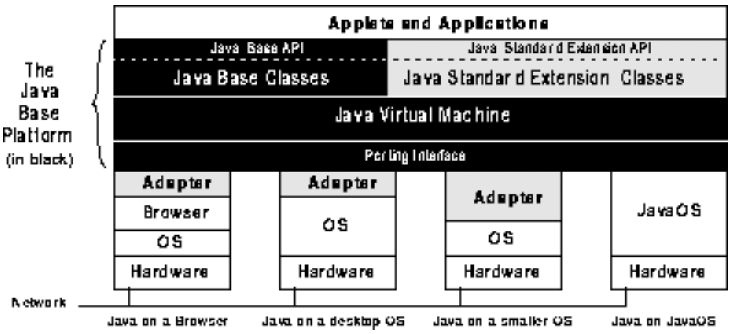
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA

Introducción

- Empresa: Sun Microsystems (comprada por Oracle).
- Origen:
  - 1990: Software para dispositivos electrónicos. James Gosling. WWW.
  - 1991: Objetivo: plataforma neutral, se empieza con C++, que se amplia y se convierte en el lenguaje Oak.
  - 1992: Oak se denomina Java.
  - 1994: Popularidad del Web. Navegador en Java (WebRunner).
  - 1995: Navegador HotJava. Netscape integra Java. JDK 1.0.
- Características:
  - Lenguaje fácil de aprender. Potente. Basado en C++ pero quitando características que contribuían a generar errores.
  - Implementa los conceptos fundamentales del paradigma de Programación Orientado a Objetos: clases, herencia, asociación, clases abstractas, polimorfismo, encapsulación, ocultamiento de información.
  - Permite el desarrollo de aplicaciones en red, distribuidas y concurrentes.

Plataforma Java

- Máquina Virtual JAVA.
- API JAVA: API (Application Programming Interface) define las funciones que implementa una librería. El API Java está formado por los métodos que proporcionan los interfaces y clases Java. El API es la base para que Java sea un lenguaje multiplataforma. Se compone de:
  - *Base API (Core API)*
  - *Standard Extension API*



Aplicaciones y Applets

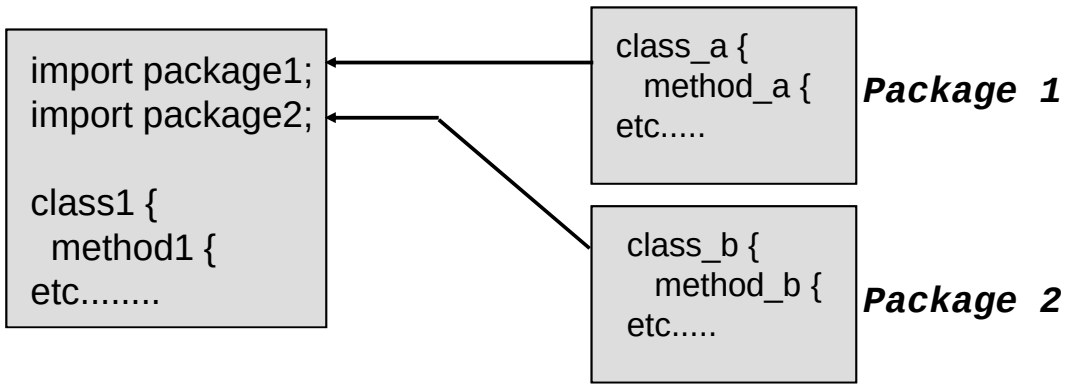
- **Aplicación:**
  - Similar a una aplicación en cualquier otro lenguaje.
  - De tipo texto o con componentes visuales.
  - Puede utilizar o no Internet.
  - Reside en la máquina donde se ejecuta.

- **Applet:**
  - Pequeño programa que necesita un navegador para su ejecución.
  - Se utiliza en el contexto de las páginas HTML.
  - Dentro de una página HTML se referencian con la palabra reservada <APPLET>.

Java visto desde el interior

- Librería de clases agrupadas en paquetes.
- Un compilador (javac.exe).
- Un entorno de ejecución o máquina virtual (java.exe).

Los paquetes favorecen la reusabilidad del código: al importar un paquete en un programa Java, se incorpora un conjunto de clases del paquete.



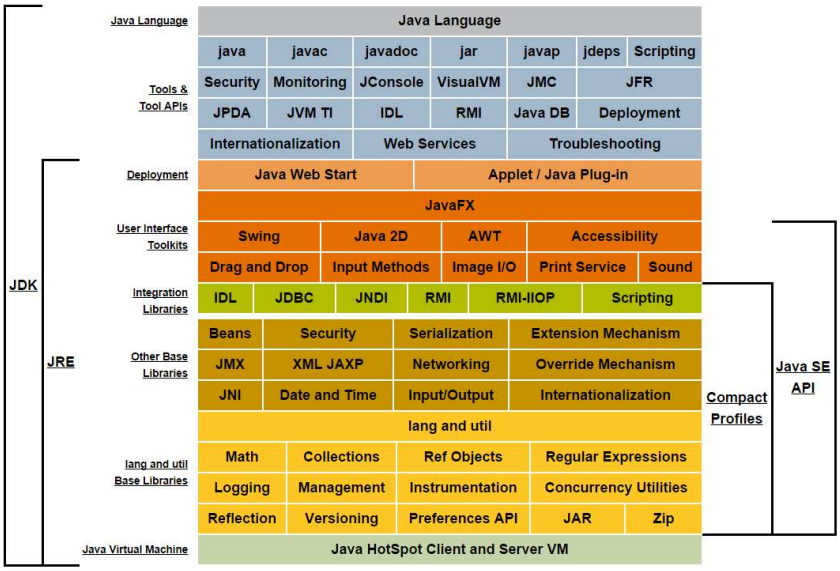
Java visto desde el interior

- **Paquetes de Java más comunes:**
  - java.lang: clases de base del lenguaje
  - java.io: clases de entrada/salida
  - java.awt / javax.swing: interfaz gráfica
  - java.util: utilidades
  - java.net: clases para la red

El desarrollador puede organizar sus propias clases en paquetes.

Para desarrollar en Java...

- Kit de desarrollo Java:
  - JDK (Java Development Kit) + documentación.
  - Editor de Texto.



... entornos de desarrollo

- Eclipse
- IntelliJ IDEA
- NetBeans (Oracle)
- JDeveloper (Oracle), …

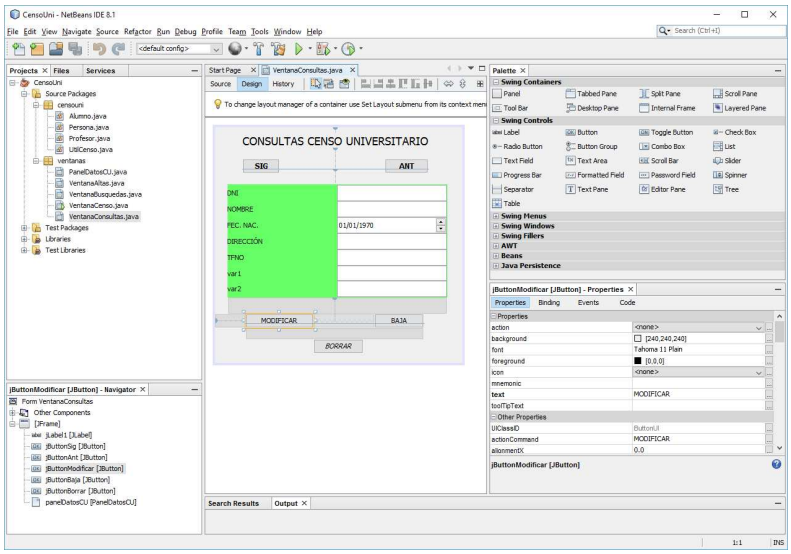


Imagen NetBeans