Universidad Autónoma del Estado de México
Centro Universitario UAEM Texcoco

Departamento de Ciencias Aplicadas.

Ingeniería en Computación.

Lenguaje de Programación Orientado a Objetos.

Unidad de competencia I: “Plataforma y lenguaje”

Presenta:

M. en C. C. J. Jair Vázquez Palma.

Lenguaje de Programación Orientado a Objetos

Objetivo de la Unidad de la Unidad de Aprendizaje

Familiarizar al alumno con un lenguaje de programación orientado a objetos, los patrones de diseño en las que se sustentan algunas de las clases en interfaces de su API, así como con una plataforma de desarrollo, que le permita la aplicación de ésta para el desarrollo de software de forma eficiente y efectiva.

Unidad I: Plataforma y lenguaje.

Contenido:

- Clasificaciones.
- Características de lo lenguajes de programación orientado a objetos.
- Características de la plataforma.
- Instalación y preparación del lenguaje y el IDE.

Objetivos de la Unidad de Competencia I

- Conocer las características de los Lenguajes de Programación Orientados a Objetos.
- Instalación del lenguaje de programación.
- Preparación de variables de entorno y configuración del IDE.
- Conocimiento de los entornos de desarrollo para Java.

Introducción a POO

- La programación OO empezó hace 30 aprox. años
- En los 1990s se incrementó dramáticamente la demanda para sistemas de software OO, por la promesa en la revolución en el desarrollo de software.
- Han aparecido varias metodologías para el desarrollo de software, que tienen que ver con algunas o todas las fases del ciclo de vida del software, desde los requerimientos al mantenimiento.

Introducción a POO

Algunas características importantes de los sistemas de software actuales son:

•*Complejidad*: la arquitectura interna de los sistemas actuales de software es compleja, incluyen frecuentemente concurrencia y paralelismo. La abstracción en términos de conceptos de OO es una técnica que ayuda a tratar con la complejidad.

•*Amigabilidad*: Este es un requerimiento de suma importancia para los sistemas de software en general.

•*Reusabilidad*: Tomar componentes creados por otros es mejor que crearlos nuevos. La herencia es un mecanismo de OO que estimula la reusabilidad del software. Facilita el rápido desarrollo del software.

Introducción POO

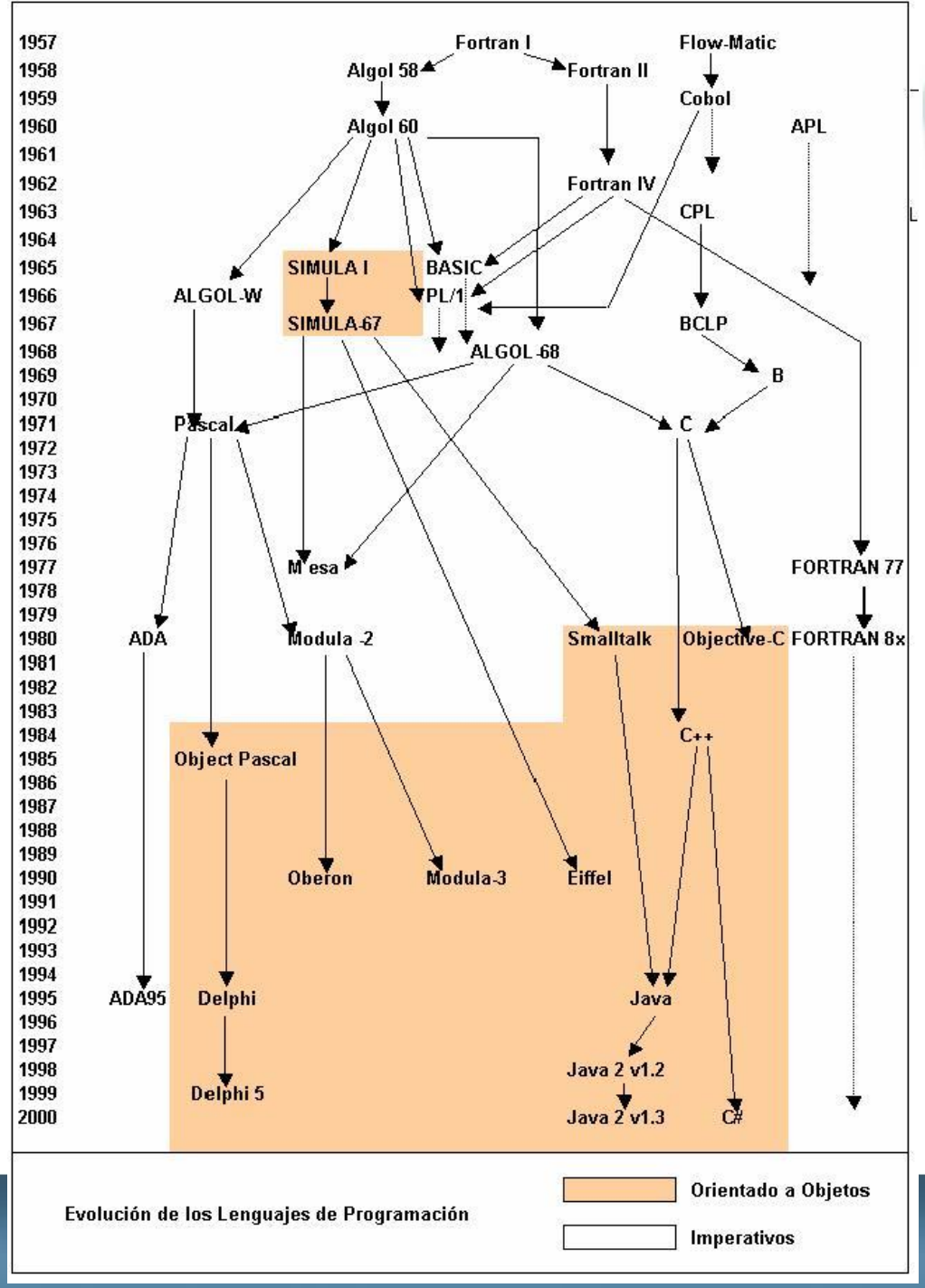
Las razones del rápido desarrollo en los últimos 15 años han sido:

- Una mejor modelación de aplicaciones del mundo real
- La posibilidad del reusó del software durante el desarrollo de un sistema de software

Características:

- OO se define por herencia, encapsulación, métodos y mensajes, como en Smalltalk.
- OO se define encapsulación, abstracción de datos, métodos, mensajes, herencia y vinculación dinámica.
- Es un modelo de simula el comportamiento ya sea de una parte del mundo real o imaginario.
- Objetos, clases y herencia. Los objetos son entidades autónomas que tienen un estado y responden a mensajes. Las clases agrupan los objetos por sus atributos y operaciones.
- Todas tienen el común de usar objetos como una encapsulación para proteger los datos con todas las operaciones legales que actúan sobre esa información oculta.

Evolución de los lenguajes de programación



Paradigma de programación:

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.

Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación, según nuestro interés de estudio.

No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Paradigma de programación:

- Hay multitud de ellos atendiendo a alguna particularidad metodológica o funcional cuando un lenguaje refleja bien un paradigma particular, se dice que soporta el paradigma, y en la práctica un lenguaje que soporta correctamente un paradigma, es difícil distinguirlo del propio paradigma, por lo que se identifica con él.

Paradigmas de programación:

SEGÚN SU NIVEL DE ABSTRACCIÓN

LENGUAJES DE BAJO NIVEL

Son aquellos que se acercan al funcionamiento de una computadora:

✓LENGUAJES MÁQUINA: ordena a la máquina las operaciones fundamentales para su funcionamiento

✓LENGUAJES ENSAMBLADORES: Con la aparición de este lenguaje se crearon los programas traductores para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina

LENGUAJES DE MEDIO NIVEL

Tienen características que los acercan a los lenguajes de bajo nivel pero, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.

LENGUAJES DE ALTO NIVEL

Se tratan de lenguajes independientes de la arquitectura del ordenador

Paradigmas de programación:

SEGÚN LA FORMA DE EJECUCIÓN

LENGUAJES COMPILADOS

Los compiladores son aquellos cuya función es traducir un programa escrito en un determinado lenguaje a un idioma que la computadora entienda. Un programa escrito en un “lenguaje compilado” se traduce a través de un programa anexo llamado “compilador” que a su vez crea un nuevo archivo independiente que no necesita ningún otro programa para ejecutarse así mismo. Este archivo se llama ejecutable.

LENGUAJES INTERPRETADOS

Un programa escrito en lenguaje interpretado requiere de un programa auxiliar (el interprete) que traduce los comando de los programas según sea necesario. Es aquel en que las instrucciones se traducen o interpretan una a una en tiempo de ejecución a un lenguaje intermedio lenguaje maquina o a través de una maquina virtual, siendo típicamente mas lentos que los programas compilados

LENGUAJES INTERMEDIARIOS

Algunos lenguajes pertenecen ambas categorías (LISP, Java, Phyton) dado que el programa escrito en estos lenguajes puede, en ciertos casos, sufrir una fase de compilación intermediaria en un archivo ininteligible (diferente al archivo fuente) y no ejecutable (requerirá un interprete).

Paradigmas de programación:

SEGÚN LA FORMA DE EJECUCIÓN

LENGUAJES COMPILADOS

Los compiladores son aquellos cuya función es traducir un programa escrito en un determinado lenguaje a un idioma que la computadora entienda. Un programa escrito en un “lenguaje compilado” se traduce a través de un programa anexo llamado “compilador” que a su vez crea un nuevo archivo independiente que no necesita ningún otro programa para ejecutarse así mismo. Este archivo se llama ejecutable.

LENGUAJES INTERPRETADOS

Un programa escrito en lenguaje interpretado requiere de un programa auxiliar (el interprete) que traduce los comando de los programas según sea necesario. Es aquel en que las instrucciones se traducen o interpretan una a una en tiempo de ejecución a un lenguaje intermedio lenguaje maquina o a través de una maquina virtual, siendo típicamente mas lentos que los programas compilados

LENGUAJES INTERMEDIARIOS

Algunos lenguajes pertenecen ambas categorías (LISP, Java, Phyton) dado que el programa escrito en estos lenguajes puede, en ciertos casos, sufrir una fase de compilación intermediaria en un archivo ininteligible (diferente al archivo fuente) y no ejecutable (requerirá un interprete).

Paradigmas de programación:

SEGÚN EL PARADIGMA DE PROGRAMACIÓN

LENGUAJES IMPERATIVOS

Aquellos en los cuales se le ordena a la computadora cómo realizar una tarea siguiendo una serie de pasos o instrucciones

LENGUAJES ORIENTADOS A OBJETOS

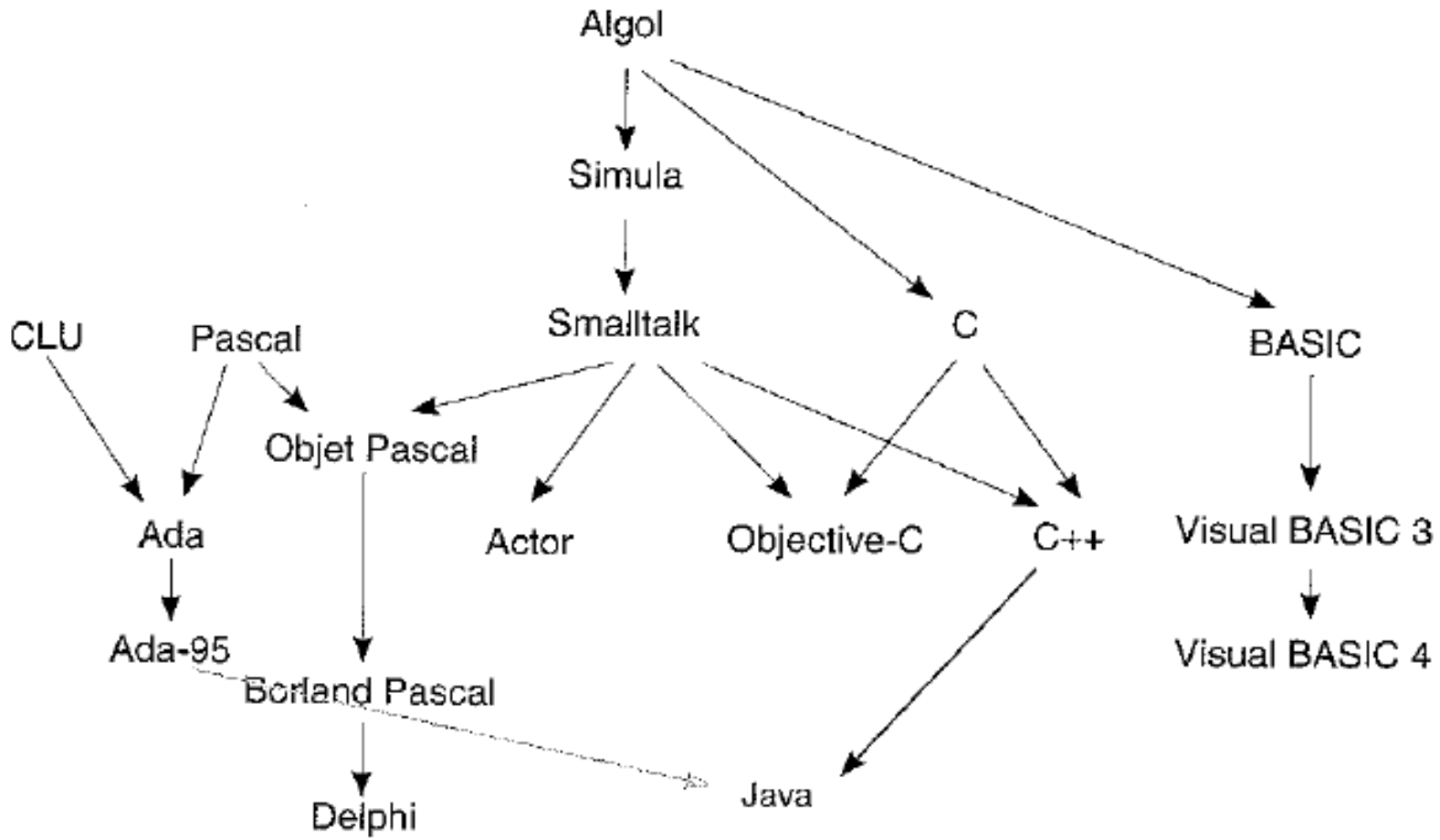
La programación orientada a objetos expresa un programa como un conjunto de objetos, que colaboran para realizar tareas.

LENGUAJES DECLARATIVOS

Aquellos en los cuales se le indica a la computadora qué es lo que se desea obtener o qué es lo que se esta buscando

- LÓGICOS
- FUNCIONALES

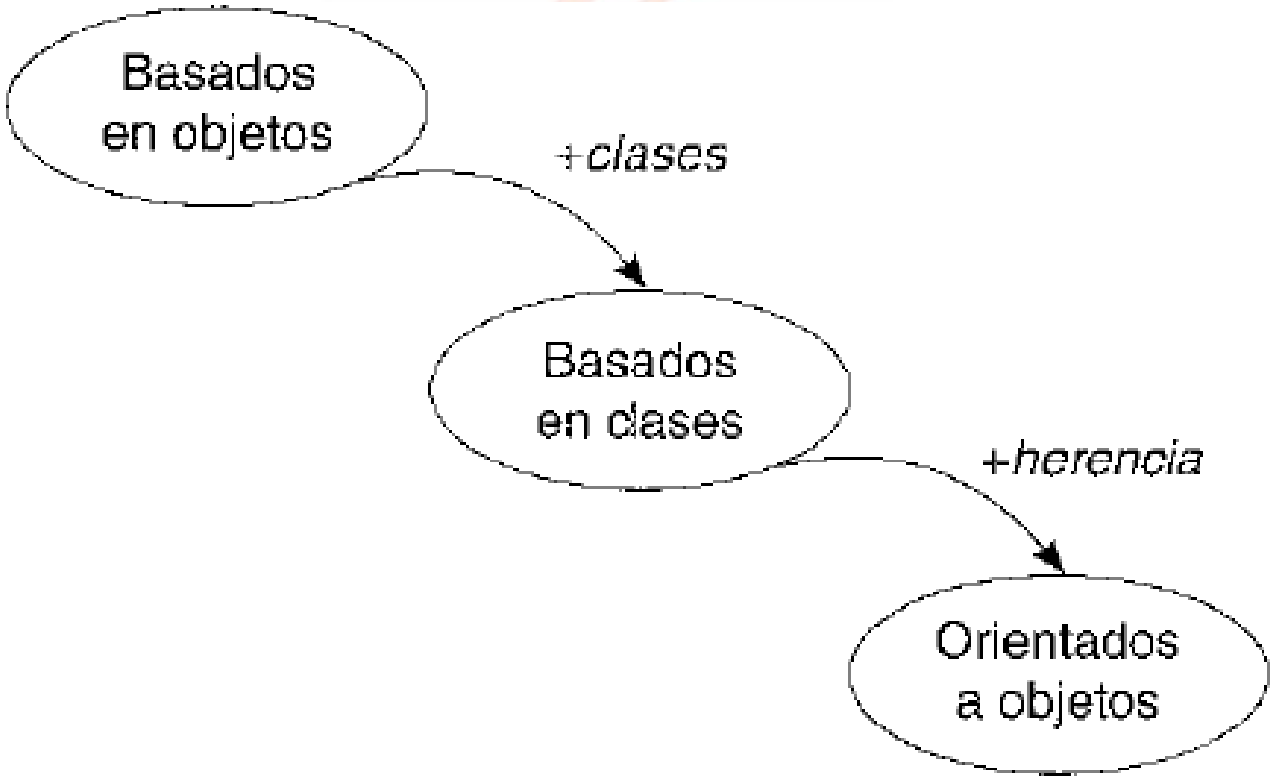
Lenguajes de programación OO



Evolución de los lenguajes orientados a objetos.

Clasificación de los lenguajes de programación OO

- Existen varias clasificaciones de POO, una de ellas ampliamente aceptada y difundida es la dada por *Wegner:*



Clasificación de lenguajes OO de *Wagner.*

Clasificación de los lenguajes de programación OO

Clasificación de lenguajes OO de *Wagner.*

- ***Lenguajes basados en objetos*** que soportan objetos. Es decir, disponen de componentes caracterizados por un conjunto de operaciones(comportamiento) y un estado.

- ***Lenguajes basados en clases*** que implican objetos y clases. Es decir disponen de componentes tipo clase con operaciones y un estado.

- ***Lenguajes orientados a objetos*** que además de objetos y clases ofrecen mecanismos de herencia entre clases. Esto es la posibilidad de derivar operaciones y atributos de una clase (superclase) a sus subclases.

Clasificación de los lenguajes de programación OO

Hoy en día es posible ampliar esa clasificación de acuerdo a criterios puramente técnicos y hacer una nueva clasificación de la categoría LOO:

Lenguajes orientados a objetos puros. Soportan en su totalidad el paradigma de orientación a objetos:

Smalltalk, Eiffel, Simula.

Lenguajes orientados a objetos híbridos. Soportan en su totalidad el paradigma de orientación a objetos sobre un núcleo de lenguaje hibrido:

- C++ (extensión de C: Borland C++, Microsoft C++, Turbo C++, Visual C++, Symantec, Watcom).
- Objective-C (extensión de C).
- Object COBOL (extensión de COBOL).
- Object Pascal (extensión de Pascal: Turbo/Borland Pascal).
- Visual Object (extensión de Clipper).
- Delphi (extensión de Turbo Pascal 7.0).
- Java (extensión de C++ y Ada-95)

Lenguajes de programación OO

Los principales lenguajes de programación orientados a objetos son:

Ada, C++, C#, VB.NET, Clarion, Delphi, Eiffel, Java, Lexico (en castellano), Objective-C, Ocaml, Oz, PHP, PowerBuilder, Python, Ruby y Smalltalk.

Comparación de algunos lenguajes de programación OO

	Criterios	Ada-83	Ada-95	C++	Eiffel	Smalltalk	Java
1.	Modularización	Sí	Sí	Sí	Sí	Sí	Sí
2	Tipos abstractos de datos	Sí	Sí	Sí	Sí	Sí	Sí
3	Gestión automática de memoria	Sí	Sí	*En parte*	Sí	Sí	Sí
4	Sólo clases	Sí	Sí	*En parte*	Sí	Sí	Sí
5	Herencia	No	Sí	Sí	Sí	Sí	Sí
6	Polimorfismo (y ligadura dinámica)	No	Sí	Sí	Sí	Sí	Sí
7.	Herencia múltiple y repetida	No	No	Sí	Sí	No	No

Criterios de *Meyer* en lenguajes OO y basados en objetos.

Clasificación de las metodologías OO

Varios métodos han sido propuestos para sistematizar el proceso de vida del software. Y muchas metodologías de desarrollo de software han sido propuestas, y éstas pueden clasificarse en tres categorías:

- Descomposición funcional.
- Énfasis en datos, más que en funciones.
- Ambos puntos de vista: funcional y datos.

Clasificación de las metodologías OO

- Descomposición funcional:

Diseño Estructural, Refinamiento por Pasos.

- Énfasis en datos, más que en funciones:

Programación Estructurada, Modelo Entidad-Relación

- Ambos puntos de vista: funcional y datos:

Análisis Estructural, Análisis de Sistemas Estructurados, Análisis de Sistemas Estructurados y Metodología de Diseño.

Metodologías OO

Metodologías “orientadas a objetos”:

- **Adaptación:** mezclar una aproximación orientada a objetos con una metodología bien conocida de desarrollo estructural.

- **Asimilación:** usar una metodología orientada a objetos para desarrollar sistemas de software, pero que siguen el modelo tradicional del ciclo de vida del software.

Metodologías OO

- La orientación a objetos tiene la necesidad de una vista organizada y manejable del desarrollo del software en todas las fases del modelo del ciclo de vida del software.

- Esta demanda ha sido satisfecha por el Lenguaje de Modelado Unificado (UML) y por herramientas CASE tales como Rational Rose.

Conclusiones:

- La programación OO tuvo un crecimiento muy grande en los 1990s
- Aunque todavía se debate los beneficios de OO, a treinta años de su invención, OO todavía seguirá usándose
- Para usar OO deben aplicarse las técnicas de Ingeniería de Software.
- Deben dominarse los tres paradigmas: alto nivel, OO, procedural.

CARACTERISTICAS DE LA PLAMAFORMA

El lenguaje de programación *Java* es un lenguaje moderno, presentado por primera vez por *Sun Microsystems* en el segundo semestre de 1995.

Aspectos que hacen de Java un lenguaje de programación cómodo y agradable:

1) LA sencillez y elegancia de cómo se escriben los programas en Java
2) Lenguaje orientado a objetos que evita muchas preocupaciones a programadores.
3) En el proceso de compilación se realizan multitud de comprobaciones que permiten eliminar muchos posibles errores posteriores.
4) Las bibliotecas ya definidas que proporciona el lenguaje para que el programador pueda utilizar sin tener que crearlas nuevamente.

La evolución de Java ha sido muy rápida, desde que se hizo público el lenguaje y un primer entorno de desarrollo, el JDK (Jva Development Kit) y el cual actualmente ha ido creciendo.

CARACTERISTICAS DE LA PLAMAFORMA

La plataforma de desarrollo de Java consta de los siguiente:

a) Lenguaje de programación: Java.

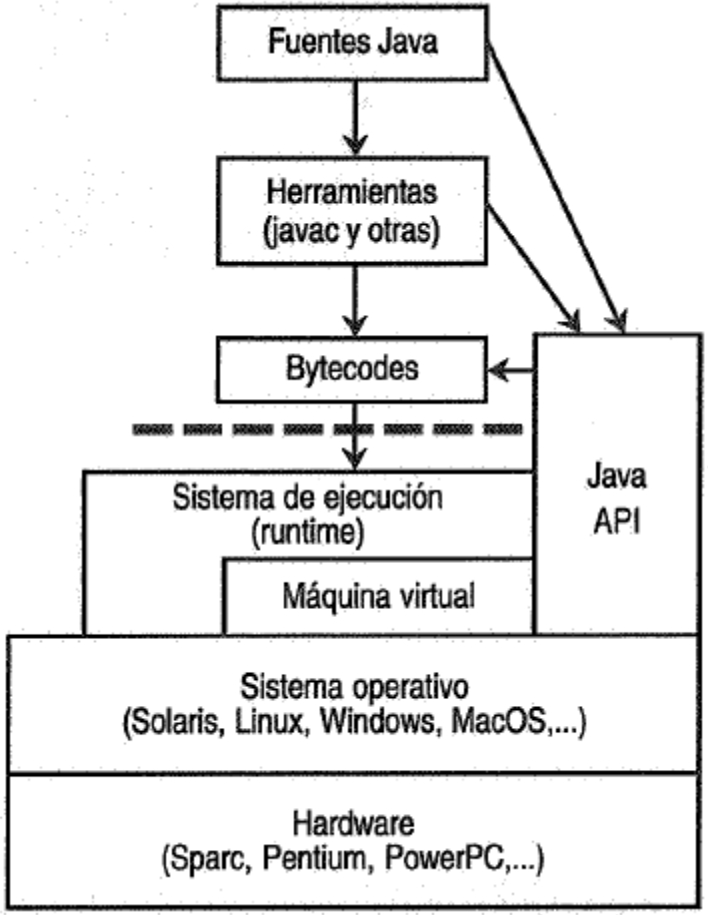
b) Un conjunto de bibliotecas estándar que se incluyen con la plataforma y que deben de existir en cualquier entorno Java.

c) Un conjunto de herramientas para el desarrollo de programas (compilador Java a código bytes, generador de documentación, depurador de programas en Java, etc.).

d) Un entorno de ejecución cuyo principal componente es una Maquina Virtual de los programas en código de bytes.

CARACTERISTICAS DE LA PLAMAFORMA

Elementos de la plataforma Java.



KIT DE DESARROLLO

Java es un tecnología que sirve para hacer aplicaciones, virtualmente, para cualquier componente que tenga un procesador de *software.*

La plataforma para el desarrollo de Java está dividida en tres ediciones:

- la estándar(JSE),
- la empresarial (JEE)
- y la de dispositivos móviles (JME).

La primera contiene, entre muchas otras cosas, **los elementos del lenguaje, los objetos para las interfaces gráficas y los mecanismos de conexión a base de datos, que son lo primero que debe** saberse para desarrollar aplicaciones Java básicas.

JRE y JDK

Sun Microsystems ofrece dos productos de software principales de la Plataforma Java ™, de la familia Standard Edition (Java ™ SE) :

Java SE Runtime Environment (JRE)
El JRE proporciona las bibliotecas, la máquina virtual de Java, y otros componentes necesarios para ejecutar applets y aplicaciones escritas en el lenguaje de programación Java. Este **entorno de tiempo de ejecución** se puede redistribuirse con las aplicaciones para hacerlos independiente.

Java SE Development Kit (JDK)
El JDK incluye el JRE de más herramientas de desarrollo de comandos en línea, tales como compiladores y depuradores que son necesarias o útiles para el desarrollo de applets y aplicaciones.

Primera parte. Iniciación al lenguaje Java

Instalación

Java funciona mediante un *software conocido como la máquina virtual (JVM por sus* siglas en inglés), que es el corazón del entorno de ejecución y que debe estar instalado en el sistema operativo para que las aplicaciones Java se ejecuten. En Windows está en una ruta semejante a esta:

C:\Program Files\Java\jre[versión]

En la que la versión es una sucesión de números como 1.4.5.6 , 1.6.0_07, etc. Para que los programas puedan ejecutarse hay que asegurarse que el JRE (*java runtime enviroment) esté instalado (que es lo más probable); si no, se descarga de:*

www.java.com

Que es un sitio dedicado exclusivamente a la disponibilidad de la máquina virtual de Java.

Para programar es necesario el *kit de desarrollo (JDK) que sirve para crear y probar las* aplicaciones y que ya incluye el JRE. La *URL de descarga es:*

http://java.sun.com/javase/downloads/index.jsp

En la que debe elegirse la presentación sencilla que dice:
Java SE Development Kit (JDK)
JDK Update [versión]
En la que, de la misma manera, versión es un número como 12, 14, 15, etc.

Hay otras descargas que tienen JFX, JEE o NetBeans pero son más grandes y los complementos que ofrecen no son necesarios para los objetivos de este curso, aunque cualquiera de ellas sirve igual.

Entorno de desarrollo integrado (EDI).

(Integrated develpment enviroment - IDE)

Contiene un editor para crear o editar el programa, un compilador para traducirlo y verificar errores de sintaxis, un *loader* para cargar los códigos, objetos de los recursos de la biblioteca utilizadas y un programa para ejecutar el programa específico.

Herramienta de desarrollo.

Entornos de desarrollo integrado populares:

BlueJ	www.blueJ.com
NetBeans	www.netbeans.org
Jbuilder	www.borland.com
Eclipse	www.eclipse.org
JCreator	www.jcreator.com
JEdit	www.jedit.org
JGrasp	www.jgrasp.org

ECLIPSE (IDE).

(Integrated develpment enviroment)

Es un entorno de desarrollo integrado multiplataforma de código abierto de amplio uso y popular; esta herramienta gratuita se encuentra disponible en www.eclipse.org y esta escrito en Java.

Eclipse fue desarrollado inicialmente por IBM como sucesor de la popular familia Visual Age; actualmente lo desarrolla la fundación *Eclipse,* organización sin fines de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios y servicios que ofrecen herramientas para numerosos lenguajes de programación, es como C, C++, PHP, Java, etcétera.

Existen dos productos de Eclipse para desarrollo en Java: *Eclipse IDE for Java EE Developers (para entornos profesionales y empresariales) y Eclipse IDE for Java SE Developers (para entorno de programación en escritorios y servidores).*

NetBeans (IDE).

(Integrated develpment enviroment)

Es un entorno de desarrollo integrado para Java; SunMicrosystem creó su propio proyecto de código abierto en Junio del 2000, y hasta hoy en día pertenece a la comunidad NetBeans.

Este entorno permite escribir, compilar, depurar y ejecutar programas. Se escribió en Java pero sirva para cualquier otro lenguaje de programación. En la actualidad existen dos productos: 1) NetBeans IDE, producto sin restricciones de uso y 2) NetBeans Platform, base modular y extensible que se emplea como estructura de integración para crear aplicaciones de escritorio.

La gran ventaja de NetBeans es su comunidad y el gran número de empresas independientes especializadas en desarrollo de Software que proporcionan extensiones adicionales, las cuales integran fácilmente en la plataforma y que también pueden utilizarse para desarrollar sus propias herramientas y soluciones.

Los dos producto de NetBeans son de código abierto y gratuito para uso tanto comercial como no comercial y cuyo código fuente este disponible para su reutilización en su sitio web.

Bibliografía:

- Joyanes A. L., Zahonero M. I., "Programación en Java 6, Algoritmos, programación orientada a objetos e interfaz gráfica de usuario", 2011, 1a edición, Ed:McGrawHill.

- Joyanes Aguilar L. Programación orientada a objetos, 1a edición, Ed:McGrawHill.

- Roger S. Pressman, Ingeniería del Software, un enfoque practico. 5a edición, Ed: McGrawHill.

- Ian Sommerville, Ingeniería del Software. 7a edición, Ed: Pearson.