	MEDIA TÉCNICA DESARROLLO DE SOFTWARE GUIA DE APRENDIZAJE # 1 Modulo Elementos de software 1 TEMA: Introducción a java	

DOCENTE: Juan Carlos Pérez Pérez
ESTUDIANTE : _____
FECHA:__/__/____ Nota:____

1. INTRODUCCIÓN A JAVA

Java surgió en 1991 cuando un grupo de ingenieros de Sun Microsystems trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.

Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollaron un código "neutro" que no dependía del tipo de electrodoméstico, el cual se ejecutaba sobre una "máquina hipotética o virtual" denominada Java Virtual Machine (JVM). Era la JVM quien interpretaba el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: "**Write Once, Run Everywhere**".

A pesar de los esfuerzos realizados por sus creadores, ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje.
Como lenguaje de programación para computadores, Java se introdujo a finales de 1995. La clave fue la incorporación de un intérprete Java en la versión 2.0 del programa Netscape Navigator, produciendo una verdadera revolución en Internet. Java 1.1 apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje. Java 1.2, más tarde rebautizado como Java 2, nació a finales de 1998.

Al programar en Java no se parte de cero. Cualquier aplicación que se desarrolle "cuelga" (o se apoya, según como se quiera ver) en un gran número de clases preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el API o Application Programming Interface de Java). Java incorpora en el propio lenguaje muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso muchos expertos opinan que Java es el lenguaje ideal para aprender la informática moderna, porque incorpora todos
estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo.

El principal objetivo del lenguaje Java es llegar a ser el “nexo universal” que conecte a los usuarios con la información, esté ésta situada en el ordenador local, en un servidor de Web, en una base de datos o en cualquier otro lugar.

Java es un lenguaje muy completo .

Por ello, conviene aprenderlo de modo iterativo: primero una visión muy general, que se va refinando en sucesivas iteraciones. Una forma de hacerlo es empezar con un ejemplo completo en el que ya aparecen algunas de las características más importantes.

La compañía Sun describe el lenguaje Java como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”. Además de una serie de halagos por parte de Sun hacia su propia criatura, el hecho es que todo ello describe bastante bien el lenguaje Java, aunque en algunas de esas características el lenguaje sea todavía bastante mejorable.

QUÉ ES JAVA 2

Java 2 (antes llamado Java 1.2 o JDK 1.2) es la tercera versión importante del lenguaje de programación Java.

No hay cambios conceptuales importantes respecto a Java 1.1 (en Java 1.1 sí los hubo respecto a Java 1.0), sino extensiones y ampliaciones, lo cual hace que a muchos efectos –por ejemplo, para esta introducción- sea casi lo mismo trabajar con Java 1.1 o con Java 1.2.

Los programas desarrollados en Java presentan diversas ventajas frente a los desarrollados en otros lenguajes como C/C++. La ejecución de programas en Java tiene muchas posibilidades:

ejecución como aplicación independiente (Stand-alone Application), ejecución como applet, ejecución como servlet, etc. Un applet es una aplicación especial que se ejecuta dentro de un navegador o browser (por ejemplo Netscape Navigator o Internet Explorer) al cargar una página HTML desde un servidor Web. El applet se descarga desde el servidor y no requiere instalación en el ordenador donde se encuentra el browser. Un servlet es una aplicación sin interface gráfica que se ejecuta en un servidor de Internet. La ejecución como aplicación independiente es análoga a los programas desarrollados con otros lenguajes.

Además de incorporar la ejecución como Applet, Java permite fácilmente el desarrollo tanto de arquitecturas cliente-servidor como de aplicaciones distribuidas, consistentes en crear aplicaciones capaces de conectarse a otros ordenadores y ejecutar tareas en varios ordenadores simultáneamente, repartiendo por lo tanto el trabajo. Aunque también otros lenguajes de programación permiten crear aplicaciones de este tipo, Java incorpora en su propio API estas funcionalidades.

EL ENTORNO DE DESARROLLO DE JAVA

Existen distintos programas comerciales que permiten desarrollar código Java. La compañía Sun, creadora de Java, distribuye gratuitamente el Java(tm) Development Kit (JDK). Se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java.

Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (con el denominado Debugger). Cualquier programador con un mínimo de experiencia sabe que una parte muy importante (muchas veces la mayor parte) del tiempo destinado a la elaboración de un programa se destina a la detección y corrección de errores. Existe también una versión reducida del JDK, denominada JRE (Java Runtime Environment) destinada únicamente a ejecutar código Java (no permite compilar).

Los IDEs (Integrated Development Environment), tal y como su nombre indica, son entornos de desarrollo integrados. En un mismo programa es posible escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Algunos incluyen una herramienta para realizar Debug gráficamente, frente a la versión que incorpora el JDK basada en la utilización de una consola (denominada habitualmente ventana de comandos de MS-DOS, en Windows NT/95/98/xp/7/8)

bastante difícil y pesada de utilizar. Estos entornos integrados permiten desarrollar las aplicaciones en la forma mucho más rápida, incorporando en muchos casos librerías con componentes ya desarrollados, los cuales se incorporan al proyecto o programa. Como inconvenientes se pueden señalar algunos fallos de compatibilidad entre plataformas, y ficheros resultantes de mayor tamaño que los basados en clases estándar.

1.2.1 El compilador de Java

Se trata de una de las herramientas de desarrollo incluidas en el JDK. Realiza un análisis de sintaxis del código escrito en los ficheros fuente de Java (con extensión *.java).

Si no encuentra errores en el código genera los ficheros compilados
(con extensión *.class).

En otro caso muestra la línea o líneas erróneas. En el JDK de Sun dicho compilador se llama javac.exe. Tiene numerosas opciones, algunas de las cuales varían de una versión a otra. Se aconseja consultar la documentación de la versión del JDK utilizada para obtener una información detallada de las distintas posibilidades.

1.2.2 La Java Virtual Machine

La existencia de distintos tipos de procesadores y ordenadores llevó a los ingenieros de Sun a la conclusión de que era muy importante conseguir un software que no dependiera del tipo de procesador utilizado. Se planteó la necesidad de conseguir un código capaz de ejecutarse en cualquier tipo de máquina. Una vez compilado no debería ser necesaria ninguna modificación por el hecho de cambiar de procesador o de ejecutarlo

en otra máquina. La clave consistió en desarrollar un código "neutro" el cual estuviera preparado para ser ejecutado sobre una "**máquina hipotética o virtual**", denominada Java Virtual Machine
(JVM). Es esta JVM quien interpreta este código neutro convirtiéndolo a código particular de la CPU utilizada. Se evita tener que realizar un programa diferente para cada CPU o plataforma.
La JVM es el intérprete de Java. Ejecuta los "bytecodes" (ficheros compilados con extensión *.class) creados por el compilador de Java (javac.exe). Tiene
Numerosas opciones entre las que destaca la posibilidad de utilizar el denominado JIT (Just-In-Time Compiler), que puede mejorar entre 10 y 20 veces la velocidad de ejecución de un programa.

1.2.3 Las variables PATH y CLASSPATH

El desarrollo y ejecución de aplicaciones en Java exige que las herramientas para compilar (javac.exe) y ejecutar (java.exe) se encuentren accesibles. El ordenador, desde una ventana de comandos de MS-DOS, sólo es capaz de ejecutar los programas que se encuentran en los directorios indicados en la variable PATH del ordenador (o en el directorio activo). Si se desea compilar o ejecutar código en Java, el directorio donde se encuentran estos programas (java.exe y javac.exe)
deberá encontrarse en el PATH. Tecleando PATH en una ventana de comandos de MS-DOS se muestran los nombres de directorios incluidos en dicha variable de entorno.

Java utiliza además una nueva variable de entorno denominada CLASSPATH, la cual determina dónde buscar tanto las clases o librerías de Java (el API de Java) como otras clases de usuario. A partir de la versión 1.1.4 del JDK no es necesario indicar esta variable, salvo que se desee añadir conjuntos de clases de usuario que no vengan con dicho JDK. La variable **CLASSPATH** puede incluir la ruta de directorios o ficheros *.zip o *.jar en los que se encuentren los ficheros ***.class**. En el caso de los ficheros *.zip hay que observar que los ficheros en él incluidos no deben estar comprimidos. En el caso de archivos *.jar existe una herramienta (jar.exe), incorporada en el JDK, que permite generar estos ficheros a partir de los archivos compilados ***.class. Los ficheros *.jar** son archivos comprimidos y por lo tanto ocupan menos espacio que los archivos *.class por separado o que el fichero *.zip equivalente.
Una forma general de indicar estas dos variables es crear un fichero batch de MS-DOS (*.bat) donde se indiquen los valores de dichas variables. Cada vez que se abra una ventana de MS-DOS será necesario ejecutar este fichero *.bat para asignar adecuadamente estos valores. Un posible fichero llamado jdk117.bat, podría ser como sigue:

```
set JAVAPATH=C:\Program Files\Java\jdk1.6.0_18
set PATH=.;%JAVAPATH%\bin;%PATH%
set CLASSPATH=.\;%JAVAPATH%\lib\classes.zip;%CLASSPATH%
```

lo cual sería válido en el caso de que el JDK estuviera situado en el directorio C:\Program Files\Java\jdk1.6.0_18

Si no se desea tener que ejecutar este fichero cada vez que se abre una consola de MS-DOS es
En el caso de utilizar Windows 7 se añadirá la variable PATH en el cuadro de diálogo que se abre con Inicio ->Equipo -> Propiedades -> Panel de control ->Configuración avanzada del programa ->Variables de entorno.
También es posible utilizar la opción –classpath en el momento de llamar al compilador javac.exe o al intérprete java.exe. En este caso los ficheros *.jar deben ponerse con el nombre completo en el CLASSPATH: no basta poner el PATH o directorio en el que se encuentra.

Terminología en Java

Esta es alguna de la Terminología utilizada en Java, lo cual demuestra el amplio alcance que tiene este lenguaje.

- **Applet** : Un applet es un programa que genera una ventana adicional dentro de su "Netscape Navigator" o "Internet Explorer" , esta ventana puede componerse de menús,sonido o imágenes. Otro terminología que casi siempre va en conjunción con un Applet es : **AWT** (Abstract Window Toolkit) y "SWING", ambas son API's ("Programming Interface") utilizados para formar Applets.
- **Java APIs** : Forman la base para programar en el lenguaje Java, estas clases se encuentran divididas por paquetes ("packages") que serian el Análogo de "Librerías en C". AWT ("Abstract Windows Toolkit") mencionado anteriormente es uno de los paquetes del API Java

- **Java Beans** : Esta arquitectura permite una manera de re-utilizar componentes de software que pueden ser manipulados en herramientas de desarrollo("Builder Tools"). Estos "Beans" pueden ser tan sencillos como un botón,o complejos como el acceso a una base de datos; una característica primordial de un JavaBean son los métodos (funciones) get|set
- **JFC "Java Foundation Classes" :** Forman parte del API Java , y son un juego de componentes para generar una interfase gráfica (GUI) y otros servicios que simplifican el desarrollo de aplicaciones en el Cliente ("Client Side") para Intranets e Internet.
- **JNI "Java Native Interface" :** Esto permite que código escrito en Java sea capaz de interactuar con aplicaciones escritas en C, C++.

- **JSP "Java Server Pages" :** Este es un tipo de programa Java que contiene HTML, para ejecutar un JSP se requiere de un servlet engine como : Tomcat o bien un java application server como Websphere de IBM que son capaces de ejecutar un "Java Server Pages".
- **JVM ("Java Virtual Machine"):** Este componente de Java es el ingrediente principal del logo "Write Once, Run Everywhere", cada plataforma o Sistema Operativo (Oracle,Windows,Linux,etc) debe desarrollar un "Virtual Machine", esto otorga un nivel de abstracción entre los programas escritos en Java y las diferentes plataformas, lo cual garantiza que toda aplicación escrita en Java logre ser ejecutada en todo tipo de Plataforma que soporte un "Java Virtual Machine".
- Esto en contraste con programas escritos en C y C++ en los que era necesario modificar el código fuente ("Source Code") para que el programa ejecutara en diferentes plataformas. Lo que interpreta un JVM "Virtual Machine" es Byte Code, el cual es generado al compilar cualquier programa en Java. Este Byte Code será idéntico si es producido en un ambiente Windows,Solaris,Linux..etc y como ya fue mencionado: a través del JVM para cada plataforma se producen los mismos resultados.
- **JDBC** : JDBC es un API que permite la ejecución de SQL (Structured Query Language) (lenguaje que utilizan las bases de datos). Con este API es posible accesar casi toda fuente de Información desde Bases de Datos, Hojas de Cálculo hasta archivos comunes("flat files")
- **JDK o SDK**: JDK es el ambiente en el cual es posible desarrollar cualquier aplicación Java. Este ambiente o paquete incluye: El API de Java, el compilador de Java, así como el JVM "Java Virtual Machine" de la plataforma correspondiente.**NOTA:** La última versión de JDK a esta fecha es JDK1.4, sin embargo, esto también es denominado "Java 2 Platform"(J2SE) e inclusive SDK ("Standard Development Kit")
- **JINI** : Esta Tecnología permite la comunicación en Red desde aparatos domésticos(Refrigeradores,Hornos y otros dispositivos) hasta Sistemas Empresariales(Sun E10000, S80 de IBM ,etc), su gran ventaja ? Esta arquitectura permite que cada servicio (aparato o software) pueda comunicarse entre sí, PERO sin necesidad de administración humana.
- **RMI "Remote Method Invocation** : Al igual que **Jini**, RMI permite que aplicaciones en Java se comuniquen a través de Red, las aplicaciones pueden ejecutarse en computadoras en lados opuestos del mundo,PERO esto se logra mediante el Protocolo GIOP "General Inter Orb Protocol", aunque se utiliza IIOP, mejor conocido como "Inter Orb Protocol" que es una variación de GIOP operado bajo el protocolo TCP/IP .
- **Servlet** : Un servlet generalmente es utilizado para procesar formas (requisiciones de usuarios),verificar ("authenticate") usuarios, generar contenido dinámico; es muy similar a un JSP, inclusive un JSP se convierte eventualmente en un Servlet, la diferencia es que un Servlet solo contiene lenguaje Java desde el inicio, mientras que un JSP contiene Java y HTML.

Tiene competencia Java ?

Si, el lenguaje se llama **C#** y es desarrollado por Microsoft, inclusive su estructura ("syntax") es muy similar a Java. **C#** es parte de la iniciativa .NET de Microsoft,pero en si .NET es rival de J2EE mas no directamente del lenguaje Java.

Al nivel de lenguaje:**C#** vs. **Java** además de las similitudes en estructura ("syntax") poseen un ambiente de ejecución similar: en Java a través de un JVM "Java Virtual Machine" y en C# a través del denominado CLR "Common Language Runtime"; de la misma manera que el lenguaje "Java" genera ByteCode para lograr interoperabilidad de plataforma y ejecutarlo en un JVM, en C# se genera MSIL o IL "Microsoft Intermediate Language" para ejecutarlo en el CLR "Common Language Runtime".

Una nota ambiciosa y curiosa acerca de MSIL o IL "Microsoft Intermediate Language"

A través del proyecto .NET se pretende generar MSIL para TODO tipo de lenguaje (inclusive Java), esto es, se puede desarrollar una aplicación en varios lenguajes y todos estos interoperar una vez convertidos a MSIL, obviamente el poder mezclar varios lenguajes en un solo programa | aplicación es algo sin precedente, Microsoft pretende lograr la interoperabilidad ofrecida por Java (a través de MSIL) y llevarla a cabo para todo lenguaje.

Consideraciones

Al gran consorcio de empresas (principalmente Sun Microsystem's su creador y ahora adquirido por el gigante Oracle) que apoyan Java les ha llevado poco más de **5 años** y **millones de dólares** no solo desarrollar varios "JVM" estables sino lograr la aceptación en la industria, desde un punto de vista técnico estos proyectos han girado alrededor de solo un lenguaje:Java.

Microsoft seguramente tiene los fondos necesarios para un proyecto de esta magnitud,pero será posible desarrollar una solución eficientemente técnica para lograr la interoperabilidad de diversos lenguajes ?

Sin duda los proyectos **.NET | C#** y **J2EE | Java** serán los dos mayores contendientes en todo sistema de computo en la próxima década y ahora que java fué adquirido por Oracle.

Ver Anexo Mapa Conceptual Java

Tomado de : http://www.osmosislatina.com/java/basico.htm[20/03/2011 10:20 p.m]

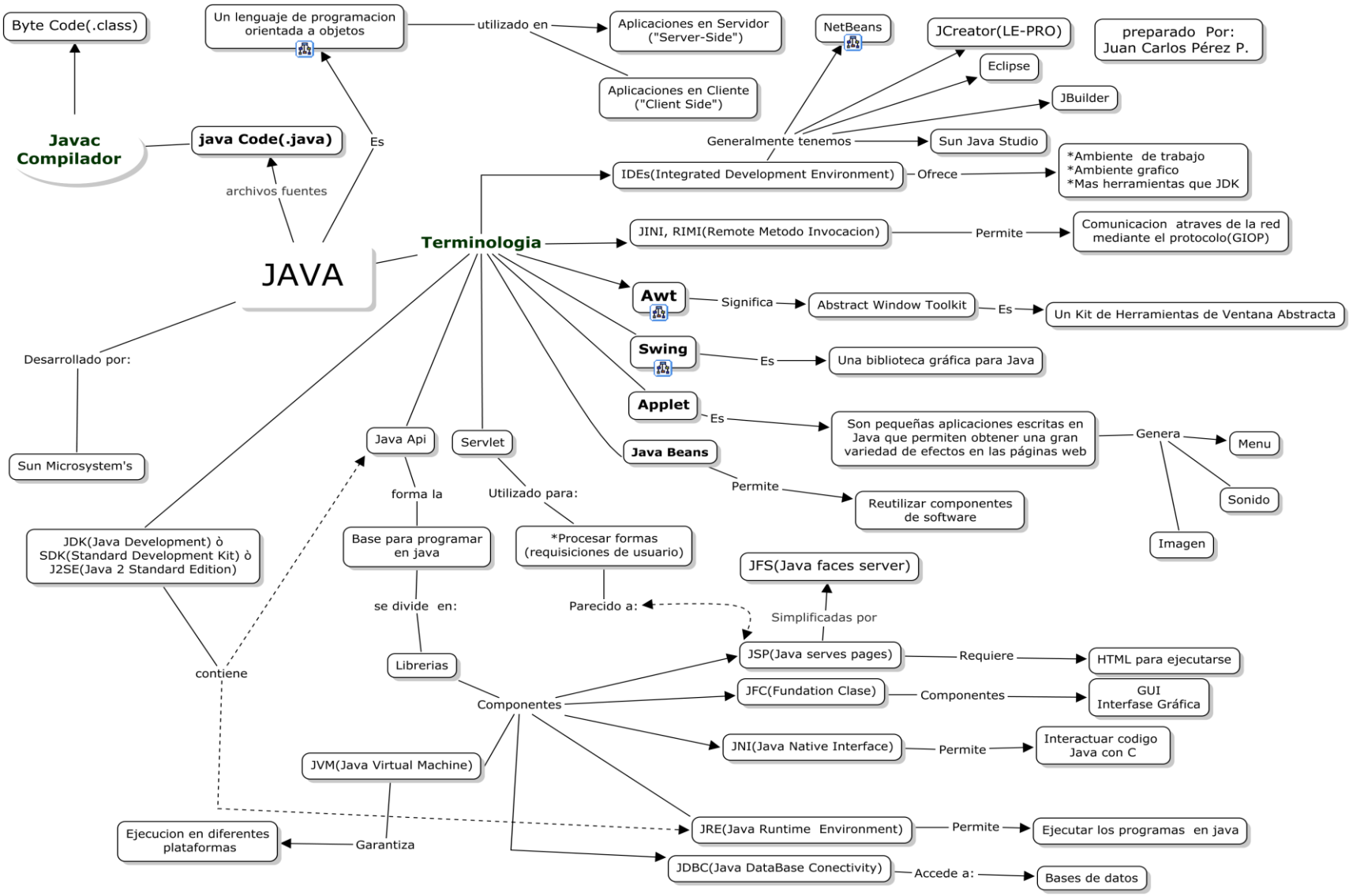
Bibliografía

DEITEL,DEILTEL Java Como programar; Séptima Edición, PEARSON Prentice Hall
CEBALLOS, Fco Javier, Java 2 Interfaces grafica y aplicaciones para Internet Editorial Alfaomega.
Oracle, http://www.oracle.com/technetwork/java/index.html

Actividad :

1.Investigue qué es la programación orientada a objetos: POO

2. Diga qué es UML


Byte Code(.class)
Un lenguaje de programacion orientada a objetos
utilizado en
Aplicaciones en Servidor ("Server-Side")
Aplicaciones en Cliente ("Client Side")
NetBeans
JCreator(LE-PRO)
Eclipse
JBuilder
Sun Java Studio
preparado Por:
Juan Carlos Pérez P.
Javac Compilador
java Code(.java)
Es
archivos fuentes
Generalmente tenemos
IDEs(Integrated Development Environment)
Ofrece
*Ambiente de trabajo
*Ambiente grafico
*Mas herramientas que JDK
JAVA
Terminologia
JINI, RIMI(Remote Metodo Invocacion)
Permite
Comunicacion atraves de la red mediante el protocolo(GIOP)
Awt
Significa
Abstract Window Toolkit
Es
Un Kit de Herramientas de Ventana Abstracta
Swing
Es
Una biblioteca gráfica para Java
Applet
Es
Son pequeñas aplicaciones escritas en Java que permiten obtener una gran variedad de efectos en las páginas web
Genera
Menu
Sonido
Imagen
Java Beans
Permite
Reutilizar componentes de software
Desarrollado por:
Sun Microsystem's
Java Api
Servlet
forma la
Utilizado para:
JDK(Java Development) ò SDK(Standard Development Kit) ò J2SE(Java 2 Standard Edition)
Base para programar en java
*Procesar formas (requisiciones de usuario)
JFS(Java faces server)
se divide en:
Parecido a:
Simplificadas por
contiene
Librerias
JSP(Java serves pages)
Requiere
HTML para ejecutarse
Componentes
JFC(Fundation Clase)
Componentes
GUI Interfase Gráfica
JNI(Java Native Interface)
Permite
Interactuar codigo Java con C
JVM(Java Virtual Machine)
JRE(Java Runtime Environment)
Permite
Ejecutar los programas en java
Ejecucion en diferentes plataformas
Garantiza
JDBC(Java DataBase Conectivity)
Accede a:
Bases de datos