UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A R A G O N

INGENIERIA EN COMPUTACION

Programación Distribuida

DESARROLLO DE APPLETS PARA EL WEB EN JAVA

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

ADRIANA MONTEJANO HERNANDEZ

ASESOR DE TESIS: ING. AMILCAR A. MONTERROSA ESCOBAR.

MARZO 1998.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DEMÉXICO
CAMPUS ARAGÓN
CARRERA DE INGENIERÍA EN COMPUTACIÓN

ING. AMILCAR A. MONTERROSA ESCOBAR

ING. ERNESTO PEÑALOZA ROMERO

ING. JUAN GASTALDI PEREZ

ING. SILVIA VEGA MUYTOY

ING. MANUEL MARTINEZ ORTIZ

Informamos a ustedes de la autorización que se le concedió a la alumna ADRIANA MONTEJANO HERNANDEZ para desarrollar el trabajo de tesis titulado "DESARROLLO DE APPLETS PARA EL WEB EN JAVA", dirigida por el Ing. AMILCAR AMADO MONTERROSA ESCOBAR solicitando a ustedes, sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar a la alumna si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro en particular, me es grato enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPÍRITU"
San Juan de Aragón, Edo. de Méx., Octubre 31 de 1996.
EL JEFE DE CARRERA

ING. JUAN GASTALDI PEREZ

JGP/srr.

AGRADECIMIENTOS

A MI MADRE

A MIS HERMANOS

Arturo, Laura y Patricia

A MIS AMIGOS

Por su amistad sincera, mucot.

LES DEDICO ESTE TRABAJO CON ESPECIAL AFECTO.

PRÓLOGO

World Wide Web es un método excepcional para distribuir información a un amplio número de personas. Las páginas Web comienzan a volverse interactivas, esto por medio de programas que crean animaciones, juegos en tiempo real, juegos para múltiples usuarios, en general todo lo que pueda realizar un pequeño programa.

El presente trabajo tiene como objetivo principal mostrar las facilidades que ofrece Java en el Web, además de ser una guía para implementar applets. Está organizado en cinco capítulos, distribuídos de la siguiente manera:

Capitulo1. INTRODUCCIÓN. Se definen conceptos básicos e historia sobre internet, www y Java, tratando aspectos generales para una mejor comprensión de los capítulos posteriores.

Capítulo 2. CARACTERÍSTICAS DE JAVA. Definiciones específicas sobre el lenguaje Java, explorando conceptos básicos de la programación orientada a objetos.

Capítulo 3. CARACTERÍSTICAS DEL PROBLEMA. Definición de un problema real, utilizado posteriormente para ilustrar la metodología de desarrollo de applets.

Capítulo 4. METODOLOGÍA DE DESARROLLO DE APPLETS. Descripción de las características de la metodología seleccionada para el diseño de applets.

Capítulo 5. CONSTRUCCIÓN DE APPLETS. Presenta el diseño de un applet, así como su interfaz. Finalizando con la integración del applet final a una página web.

CONTENIDO

Capitulo 1

Introducción

1.1 MARCO TEÓRICO

Debido a las relaciones económicas, políticas y sociales que existen entre todos los países del mundo, estos tienen una necesidad directa de comunicación. Se ha considerado, entre muchas otras opciones, utilizar la computadora para solucionar esta necesidad. Ahora tenemos acceso a información localizada en un lugar diferente gracias a la conexión de computadoras, esta conexión se denomina red.

Una red es definida en su aspecto más sencillo como :

" Una conexión de varias computadoras a través de un cableado especial para compartir recursos, datos e información"[1]

Las redes se crearon inicialmente por la necesidad de compartir datos y recursos de alto costo entre gente que trabajaba sobre los mismos o similares proyectos. Al iniciarse, solo eran conexiones aisladas de unas cuantas computadoras en un solo lugar; con el paso del tiempo crecieron, se conectaron más computadoras. Posteriormente comenzaron a conectar entre sí las pequeñas redes, así poco a poco entre todas las redes crearon una gran red denominada red de redes y se llama Internet. El uso principal que se hace de Internet e incluso de las redes internas(corporativas) es correo electrónico (e-mail), aunque actualmente hay un auge sorprendente de la navegación web.

1.2 INTERNET

1.2.1 ANTECEDENTES HISTÓRICOS

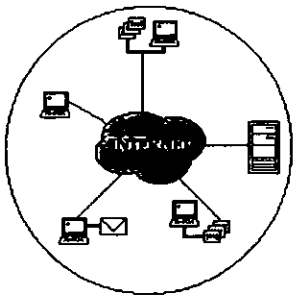
La palabra INTERNET viene de la contracción de "International - Network" , que significa red internacional. Tiene sus raíces en los años sesenta, como un proyecto de comunicaciones del departamento de Defensa de los Estados Unidos llamado ARPANET[2]. Durante esos años existía la preocupación de que una guerra pudiera cortar totalmente las comunicaciones, así que el proyecto consistía en encontrar vías flexibles para conectar redes.

Crearon ARPANET con todos los aspectos necesarios para cubrir sus necesidades, por ejemplo, hicieron estándares para la creación de direcciones y para los protocolos de comunicaciones; en la comunicación, el mensaje es puesto en un paquete IP (Internet Protocol) y es enviado por la computadora fuente. La computadora fuente es responsable de que el mensaje llegue a su destino, no la red.

En 1980 ARPANET fue dividida en dos redes, MILNET que se ocupó de asuntos militares, en tanto que ARPANET transportaba datos de investigación. Para mediados de los ochenta la Fundación Nacional de Ciencia de los Estados Unidos, se enlazó a ARPANET con seis centros de supercomputadoras en una red llamada NSFNet.

La NSF promovió el acceso universal a las instituciones educativas, financiando conexiones a las universidades, únicamente si tenían un plan para permitir el acceso en las zona. De esta manera toda persona que estuviera inscrita podía ser usuario de NSFNET. Posteriormente la red tuvo un crecimiento inesperado, hasta convertirse en la red mas grande del mundo conocida ahora como *INTERNET.*

[1] Khoshafian
[2] ARPANET (Advanced Research Project Network)



1.2.2 SERVICIOS DE INTERNET

La red mundial ofrece comunicación a lugares muy remotos en pocos segundos, ofrece múltiples servicios entre los cuales destacan telnet, ftp, correo electrónico y web, como los más importantes.

Telnet:

Es la herramienta más utilizada para accesar a los servicios de internet. Asi pues, es la conexión desde un ordenador remoto al servidor. Mediante esta aplicación, podemos adentrarnos en el sistema operativo de un ordenador remoto y permite el uso de los recursos en tiempo real.

Ftp:

Es el protocolo más utilizado en internet para transferir archivos de un lugar remoto hacia otro.

Correo electrónico:

Permite mandar y recibir "cartas electrónicas" desde cualquier parte del mundo de una forma mucho más rápida que el correo tradicional.

Existe un servicio más reciente, pero no menos importante, el World Wide Web, que se traduce como la "Gran Telaraña Mundial"; este servicio es también conocido como WWW, [3] o simplemente Web.

1.2.3 WIDE WORLD WEB

El Web se desarrollo en el Laboratorio Europeo de Partículas Físicas, conocido como el CERN, organización comprometida con la investigación sobre la física y la energía de alto nivel.

En 1989 el físico Tim Bernes-Lee propuso el Web como un sistema para transferir ideas y datos de investigación entre la comunidad de científicos relacionados con la física y la energía. El software permitía leer y transmitir documentos basados en una tecnología llamada hipertexto, denominada así por que las palabras seleccionadas en el texto pueden ser "expandidas" en cualquier momento para proporcionar más información sobre la palabra. Esto permite que las palabras formen vínculos hacia documentos que pueden contener texto, imágenes, audio o cualquier otra cosa.

Todo esto se hace por medio de programas llamados "hojeadores", también son conocidos como navegadores o browser. Hasta ahora existen dos principales tipos de navegadores : Los de solo texto y los visualizadores Web gráficos, donde se pueden ver imágenes en línea, fuentes, diseño de documentos, etc.

Los visualizadores gráficos tienen sus restricciones, si se quiere reproducir un sonido o ejecutar un programa de demostración, primero hemos de copiar el fichero en cuestión y luego utilizar un programa en nuestro ordenador capaz de entender el

formato de ese fichero, o bien cargar un módulo (plug-in) en nuestro navegador para que pueda interpretar el fichero que se ha copiado.

Analizando las características y deficiencias del Web, Sun en 1995 presentó un lenguaje de programación Orientado a Objetos llamado Java.

1.3 JAVA

1.3.1 ANTECEDENTES HISTÓRICOS

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems, líder en servidores para Internet. En un intento de resolver simultáneamente todos los problemas que se le plantean a los desarrolladores de software por la proliferación de arquitecturas incompatibles, tanto entre las diferentes maquinas como entre los diversos sistemas operativos y sistemas de ventanas que funcionaban sobre una misma maquina.

Java fue modelado a partir de C++ y su mercado fue previsto inicialmente para equipos domésticos: microondas, tostadoras, teléfonos y televisión interactiva. Este mercado, dada la falta de pericia de los usuarios para el manejo de estos dispositivos, requería unas interfaces mucho mas cómodas e intuitivas que los sistemas de ventanas que proliferaban en el momento.

James Gosling, miembro de Sun con experiencia en lenguajes de programación, decidió que las ventajas aportadas por C++ no compensaban el costo de pruebas y depuración, además el programa debe ser compilado para un chip, y si se cambia el chip, todo el software debe compilarse de nuevo, esto en el campo de la electrónica de consumo esto encarece mucho el desarrollo.

Gosling había estado trabajando en un lenguaje de programación, para el apoyo a una red de dispositivos heterogéneos de comunicación, que vendría a solucionar algunos problemas, este lenguaje fue llamado OAK, el cual, aún partiendo de la sintaxis de C++ intentaba remediar las deficiencias que iba observando.

El primer proyecto en que se aplicó este lenguaje recibió el nombre de proyecto Green y consistía en un sistema de control completo de los aparatos electrónicos. Para ello se construyó un ordenador experimental denominado *7 (Star Seven). El sistema presentaba una interfaz basada en la representación de la casa de forma animada y el control se llevaba a cabo mediante una pantalla sensible al tacto. En el sistema aparecía Duke, la actual mascota de Java. Posteriormente se aplicó a otro proyecto denominado VOD (Video On Demand) en el que se empleaba como interfaz para la televisión interactiva. Este proyecto nunca se convirtió en un sistema comercial, pero fue desarrollado en un Java enteramente primitivo. En este momento sus principales características fueron pequeñez, eficiencia y portabilidad.

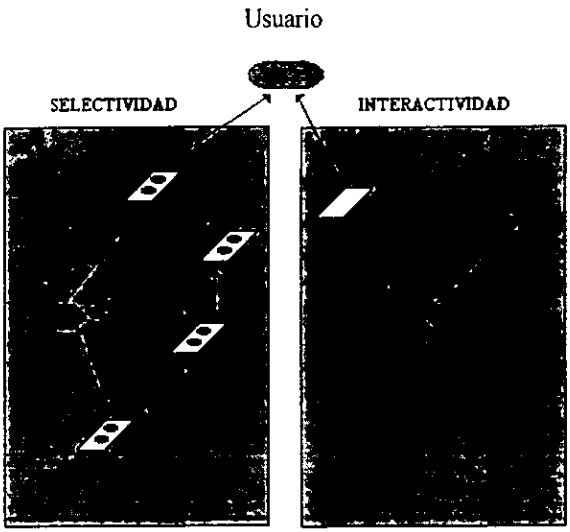
Posteriormente, analizando las características que ofrecía Oak decidieron que internet podría ser el campo de adecuado para disputar a Microsoft la primacía en el terreno del software. Oak fue renombrado como Java. En unos cuantos meses, se escribe HotJava, como vehículo para correr applets, así como también para mostrar el tipo de aplicaciones complejas que pueden ser escritas en Java. Así en mayo de 1995, en Sun World'95, presentaron formalmente a Java.

Hasta ahora, la única forma de realizar una pagina web con contenido "interactivo", era mediante la interfaz CGI (Common Gateway Interface) o scripts, que son una serie de instrucciones que indican aun programa como ejecutar un procedimiento específico, consistente en su totalidad de selectividad; esto significa que en el momento en que se accede al servidor Web, el contenido se presenta utilizando hipertexto. Los enlaces existentes en las páginas del hipertexto dan al usuario numerosas opciones de selectividad para encontrar información en una base de datos.

Sin embargo, no se ofrece información resultado de procesos ejecutados por el programa mismo.

Java ofrece un nivel de interactividad, donde un programa acepta entradas de datos provenientes del usuario a través de la página Web. Basadas en estas entradas del usuario, este contenido se procesa y da un resultado.

Por esto Java supone otro nivel de interactividad, en vez de dejar al servidor la tarea de procesar un resultado, el navegador preparado para Java es el mecanismo para procesarlo.



Con Java se puede reproducir sonido directamente desde el navegador, sin el módulo plug-in correspondiente, también se visitan páginas con animaciones, se puede enseñar al navegador a manejar nuevos formatos de ficheros, e incluso, cuando se pueda transmitir vídeo por las líneas telefónicas, nuestro navegador estará preparado

para mostrar esas imágenes. En términos generales, proporciona una forma diferente de hacer que ese navegador sea capaz de ejecutar programas.

Solamente algunos navegadores que existen en estos momentos en el mercado soportan Java, estos son :

- Netscape Navigator 2.0 en adelante. A mediados de 1995, Netscape Comunications anuncia la incorporación de la nueva generación de navegadores (Navigator 2.0 y Gold Navigator 2.0), capaces de correr applets en ellos. ofrece buen soporte para applets y en este momento es el mejor hojeador.
- Microsoft Explorer 2.0.
- Hot Java, es un navegador con soporte Java (Java-enabled), tiene como característica principal, el ser una aplicación totalmente desarrollada en lenguaje Java. Como cualquier navegador de Web, HotJava puede decodificar HTML estándar y URLs estándares, aunque no soporta completamente el estándar HTML 3.0. pero es eficaz en términos de applets. La diferencia con Netscape, es que puede escribir y leer en el disco local, aunque esto hace disminuir la seguridad. No obstante, el utilizar esta característica de HotJava es decisión del usuario.

Utilizando Java, se pueden añadir aplicaciones que vayan desde experimentos científicos interactivos de propósito educativo, a juegos o aplicaciones especializadas para la televenta. Por ejemplo, alguien podría escribir un programa Java que implementara una simulación química interactiva (una cadena de ADN).

Utilizando un navegador con soporte Java, un usuario podría recibir fácilmente esa simulación e interaccionar con ella, en lugar de conseguir simplemente un dibujo estático y algo de texto.

Además, Java proporciona una nueva forma de acceder a las aplicaciones. El software viaja transparentemente a través de la red. No hay necesidad de instalar las aplicaciones, ellas mismas vienen cuando se necesitan. Por ejemplo, la mayoría de los

navegadores del Web pueden procesar un reducido número de formatos gráficos (típicamente GIF y JPEG). Si se encuentran con otro tipo de formato, el navegador estándar no tiene capacidad para procesarlo, tendría que ser actualizado en su próxima versión para poder aprovechar las ventajas del nuevo formato. Sin embargo, un navegador con soporte Java puede enlazar con el servidor que contiene el algoritmo que procesa ese nuevo formato y mostrar la imagen. Por lo tanto, si alguien inventa un nuevo algoritmo de compresión para imágenes, el inventor sólo necesita estar seguro de que hay una copia en código Java de ese algoritmo instalada en el servidor que contiene las imágenes que quiere publicar. Es decir, los navegadores con soporte Java se actualizan a sí mismos sobre la marcha, cuando encuentran un nuevo tipo de fichero o algoritmo.

1.3.2 CARACTERÍSTICAS DE JAVA

Sun ofrece como principales características de Java :

1. SIMPLEZA. Java ofrece la potencialidad de un lenguaje como C++, pero sin las características menos usadas y más confusas. Las características principales que elimina de C++ son:

Aritmética de punteros
Registros (struct)
Definición de tipos (typedef)
Macros (#define)
Necesidad de liberar memoria (free)
Múltiple herencia

En realidad, lo que hace Java es eliminar las palabras reservadas struct y typedef, ya que las clases son parecidas. Añade características útiles como el garbage collector, reciclador de memoria dinámica. No es necesario preocuparse por liberar memoria, el reciclador se encarga de ello y como es una tarea (thread) de baja prioridad, cuando entra en acción, permite liberar bloques de memoria grandes.

Otro aspecto que hace simple a Java es su pequeño tamaño. Uno de los objetivos de Java es facilitar la construcción de software que pueda correr en máquinas pequeñas. El tamaño de un intérprete básico y soporte de clases, está alrededor de 40Kb; agregando las librerías básicas estándar y soporte de tareas suma adicionalmente 175Kb, por lo tanto su tamaño está alrededor de los 215Kb.

2. ORIENTACIÓN A OBJETOS. Java trabaja sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias de la orientación a objetos: encapsulación, polimorfismo y herencia. Las plantillas de objetos son llamadas clases, y sus copias, instancias. Estas instancias necesitan se construidas y destruidas en espacios de memoria. Incorpora funcionalidades inexistentes en C++, como por ejemplo la resolución dinámica de métodos; esta se inicia cuando un objeto llama a un método en particular, Java busca la definición de ese método en la clase de objeto, si no esta definido busca en la superclase, y así sucesivamente hacia arriba en la cadena hasta que encuentra la definición del método.

3. DISTRIBUIDO. Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que corran en varias maquinas, interactuando.

Java se ha constituido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp.

Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

4. ROBUSTEZ. Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Proporciona

Comprobación de punteros

Comprobación de límites de arrays

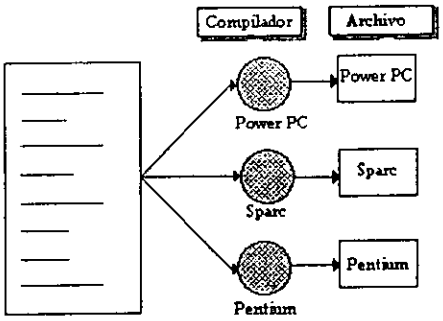
Excepciones

Verificación de archivos bytes-code

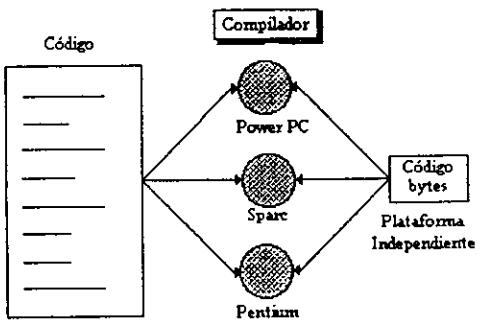
La comprobación de tipos ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo las posibilidades de error. Evita la posibilidad de sobreescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen el tiempo de desarrollo de aplicaciones.

5. ARQUITECTURA NEUTRAL. Java se diseño para soportar aplicaciones sobre redes. En general las redes son compuestas por una diversidad de sistemas con un gran variedad de CPU y arquitecturas de sistema operativo.

Normalmente cuando se compila un programa hecho en C, el compilador translada el programa hacia instrucciones de procesador. Si desea ejecutar este programa en cualquier otro sistema, se necesita tener el programa fuente, un compilador para ese sistema y recompilar el código.



Java como parte integral de la red, compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará, conocido como byte-code de Java. El código en el archivo fuente se "compila" a un código de bytes de alto nivel independiente de la máquina. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en lo más mínimo la maquina en que ha sido generado.



Este byte-code esta diseñado para ejecutarse en una máquina hipotitica que es implementada en el sistema *run-time*, que *sí es dependiente* de la máquina.

Por ejemplo, construye sus interfaces a usuario a través de un sistema abstracto de ventanas de forma que puedan ser implantadas en entornos Unix, Pc o Mac.

6. SEGURIDAD. La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros se eliminan para prevenir el acceso ilegal a la memoria. En C existen errores de alineación donde se utilizan punteros en conjunción con operaciones aritméticas. Esto permite referenciar a un lugar conocido en la memoria y poder sumar o restar algún valor para referirse a otro lugar en la memoria.

Otra laguna es el Caballo de Troya, se presenta un programa como una utilidad resultando tener una funcionalidad destructiva.

El código Java pasa muchos test antes de ejecutarse en una máquina. El código pasa a través de un verificador de byte-code que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal. Código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto. Si no se genera ningún mensaje de error, se sabe que :

El código no produce desbordamiento de operandos en la pila.

El tipo de los parámetros de todos los códigos de operación son conocidos y correctos.

No ha ocurrido ninguna conversión ilegal de datos, tal como convertir enteros en punteros.

El acceso a los campos de un objeto se sabe que es legal: public, private, protected.

No hay ningún intento de violar reglas de acceso y seguridad establecidas.

El cargador de clases también ayuda a mantener su seguridad, separando el espacio de nombres del sistema de ficheros local, de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo Caballo de Troya, ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior. Las clases importadas de la red se almacenan en un espacio de nombres privado, asociado con el origen. Cuando una clase del espacio de nombres privado accede a otra clase, primero busca en las clases predefinidas (del sistema local) y luego en el espacio de nombres de la clase que hace referencia. Esto imposibilita que una clase suplante a una predefinida.

Java imposibilita el abrir algún fichero de la máquina cliente, siempre que se realicen operaciones con archivos, éstos trabajan sobre el disco duro donde partió el applet, nadie puede utilizar nuestra máquina para hacer peticiones o realizar operaciones con otra.

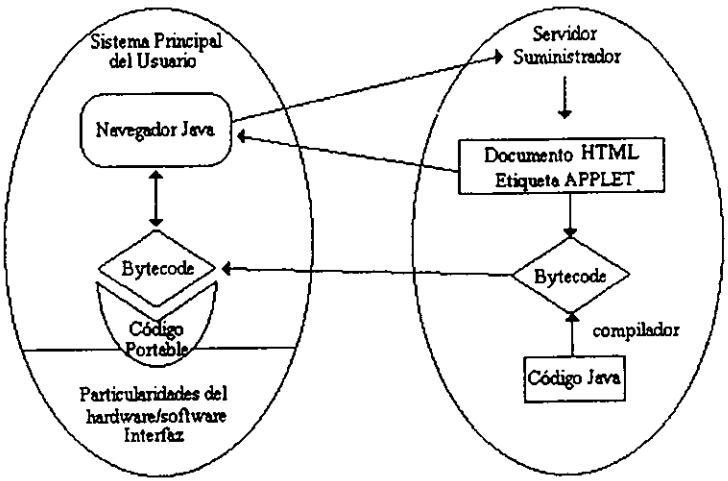
Java tiene un aspecto pendiente, en cuestión de seguridad del código fuente. JDK (Java Development Kit) proporciona un desensamblador de byte-code, que permite que cualquier programa pueda ser convertido a código fuente, lo que para el programador significa vulnerabilidad total a su código.

7. INTERPRETADO. El intérprete Java (sistema run-time) puede ejecutar directamente el codigo-objeto.

Java es mas lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en un programa tradicional.

El código fuente es escrito en cualquier editor, se compila generando el byte-code. Este código intermedio es de muy bajo nivel, pero sin alcanzar las instrucciones maquina propias de cada plataforma. El byte-code corresponde al 80% de las instrucciones de la aplicación.

Ese mismo código es el que se puede ejecutar sobre cualquier plataforma. Para ello hace falta el run-time, que es completamente dependiente de la máquina y del sistema operativo, que interpreta dinámicamente el byte-code y añade el 20% de instrucciones que faltaba para su ejecución. Con este sistema es fácil crear aplicaciones multiplataforma, aunque es necesario que exista el run-time correspondiente al sistema operativo utilizado.



8. MULTI-TAREA (MULTITHREADED).Por ser multitarea, permite muchas actividades simultáneas en un programa. Las tareas (threads), a veces llamados procesos ligeros o hilos, son básicamente pequeños procesos o piezas independientes de un gran proceso.

El beneficio de ser multitarea consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo donde se ejecute.

Por ejemplo, al navegar en el Web es frustante esperar por una gran imagen que se esta trayendo. En Java, las imágenes se pueden ir trayendo en una tarea

independiente, permitiendo que el usuario pueda acceder a la información en la página sin tener que esperar por el navegador.

9. DINÁMICO. Java conecta todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Si se ejecuta una aplicación sobre la red y encuentra una pieza de aplicación que no sabe manejar, es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar.

Para evitar que los módulos de byte-codes o los objetos de nuevas clases, haya que estar trayéndolos de la red cada vez que se necesiten, implementa opciones de persistencia, para que no se eliminen cuando se limpie la caché de la máquina.

Capitulo 2

Características de Java

2.1 Antecedentes Históricos

A principios de la década de los años 90's, las posibilidades del software estaban retrasadas respecto a las del hardware en un mínimo de dos generaciones de procesadores y la distancia continúa aumentado. Una de las preocupaciones actuales es la de crear software y sistemas corporativos más pronto y de más bajo costo. No se trata de rendimiento de hardware (la carrera por los millones de instrucciones por segundo), si no de buscar soluciones para incrementar la productividad en el desarrollo de aplicaciones. Los medios para lograr el cambio está en el uso y combinación de varias herramientas y técnicas.

La Orientación a Objetos no es un concepto nuevo, de hecho, tiene por lo menos 20 años de antigüedad. Sus raíces se encuentran en Noruega a finales de los años 60's en conexión con un lenguaje llamado Simila67, que introdujo por primera vez los conceptos de clase, corrutinas y subclases muy parecidos a los lenguajes orientados a objetos actuales. El primer lenguaje completamente orientado a objetos, lo crearon los científicos del centro científico de Palo Alto de Xerox , a mitades de la década de los 70's, y lo llamaron Smaltalk.

Existen múltiples maneras para definir la Programación Orientada a Objetos:

> " Una disciplina de ingeniería de desarrollo y modelado de software que permite construir más fácilmente sistemas complejos a partir de componentes individuales "[1]

> " Una representación directa del modelo del mundo real, transformando los requerimientos del sistema, definidos en términos del usuario, a las especificaciones del sistema, definidas en términos de la computadora"[2]

" Método de implementación en que los programas son organizados como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de una clase, cuyas clases son miembros de jerarquías de clases unidas a través de herencia"[3]

Aunque existan varias definiciones de Programación Orientación a Objetos, en general, los aspectos a considerar para una buena orientación a objetos son los mismos:

- Estructura basada en objetos
- Abstracción de datos
- Clases
- Encapsulación
- Herencia
- Polimorfismo

Si bien es cierto que las técnicas orientadas a objetos son poderosas, no proporcionan la magnitud del cambio necesario, se deben conjuntar con otras tecnologías de software. El futuro cambio, plantea software compilado a partir de componentes de objetos reutilizables, con lo que se crearía una enorme biblioteca de componentes, creando cajas negras donde no podremos mirar al interior.

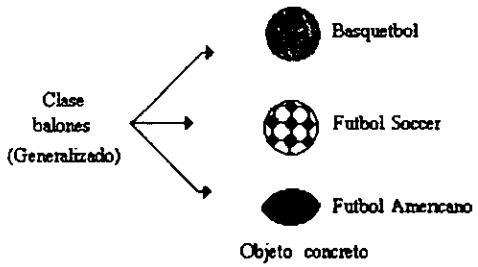
Algunos de los conceptos básicos de la Programación Orientada a Objetos, se describen a continuación.

[1] Khoshafian, S. "Object Orientation".
[2] Ledbetter, L. "Software-IC's"

2.2 CONCEPTOS BÁSICOS (POO)

1.2.1 INSTANCIA

Una instancia es otra palabra para nombrar un objeto, el cual es un conjunto abstracto de información. Un objeto toma las características propias de su clase, por lo tanto, es su representación concreta. Por ejemplo, en la clase balones, un objeto o instancia sería un balón de fútbol o de basquetbol.



1.2.2 CLASE

Una clase es una generalización de características para un tipo específico de objetos, por lo tanto, son una descripción a partir de la cual se generan objetos. Tienen dos componentes principales, atributos (variables) y comportamiento (métodos).

En C la unidad fundamental son los ficheros con código fuente, en Java son las clases. De hecho son pocas las sentencias que se pueden colocar fuera del bloque de una clase. La palabra clave import (equivalente al #include) puede colocarse al principio de un fichero, fuera del bloque de la clase. Sin embargo, el compilador reemplazara esa sentencia

[3] Booch, G. "Object -Oriented Analysis and Design with Applications"

con el contenido del fichero que se indique, que consistirá, como es de suponer, en mas clases.

Los tipos de clases que podemos definir son:

abstract

Una clase abstract tiene al menos un método abstracto. Una clase abstracta no se instancia, sino que se utiliza como clase base para la herencia.

final

Una clase final se declara como la clase que termina una cadena de herencia. No se puede heredar de una clase final.

public

Las clases public son accesibles desde otras clases, bien sea directamente o por herencia. Son accesibles dentro del mismo paquete en el que se han declarado. Para acceder desde otros paquetes, primero tienen que ser importadas.

synchronizable

Este modificador especifica que todos los métodos definidos en la clase son sincronizados, es decir, que no se puede acceder al mismo tiempo a ellos desde distintos procesos; el sistema se encarga de colocar las banderas (flag) necesarios para evitarlo. Este mecanismo hace que desde procesos diferentes se puedan modificar las mismas variables sin que haya problemas de que se sobreescriban.

2.2.3 ATRIBUTOS

Los atributos son las características individuales que diferencian un objeto de otro, y determinan la apariencia, u otras cualidades de ese objeto. Los atributos se definen con

variables; de hecho, puede considerarlos como análogos a las variables "globales" del objeto completo.

Puesto que cada objeto de una clase puede tener diferentes valores para sus variables, a cada variable se le llama variable de instancia, en general definen las características de un objeto. Cada instancia guarda su propio valor para ese atributo.

Cada atributo, cuenta con una correspondiente variable de instancia; así que, al cambiar el valor de una variable se cambia el atributo de ese objeto. Las variables de instancia pueden fijarse cuando se crea un objeto y permanecen constantes durante la vida del objeto, o bien, se puede habilitar para cambiarse.

Por ejemplo :

Moto1.Color = Negro ;

Moto1.Modelo = Harley ;

Moto2.Color = Rojo;

Moto2.Modelo = Honda ;

Donde Color y Modelo son los atributos que diferencian Moto1 de Moto2 y le dan características particulares a cada objeto.

2.2. 4 COMPORTAMIENTO

El comportamiento de clase determina que instancias de esa clase requieran cambiar su estado interno o cuando esa instancia es llamada para realizar algo por otra clase u objeto. Para definir el comportamiento de un objeto, se crean métodos, los cuales tienen un comportamiento igual al de las funciones en otros lenguajes, pero se definen dentro de una clase. Java no cuenta con funciones definidas fuera de las clases.

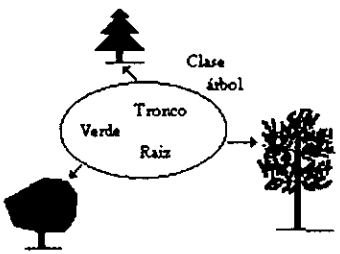
Los métodos operan sobre los objetos de esa clase; así estos objetos logran una comunicación entre sí. Una clase u objeto puede llamar a métodos en otra clase u objeto

para avisar sobre los cambios en el ambiente o para solicitarle a ese objeto que cambie su estado.

2.3 CARACTERÍSTICAS DE LOS LENGUAJES ORIENTADOS A OBJETOS

2.3.1 ENCAPSULACIÓN

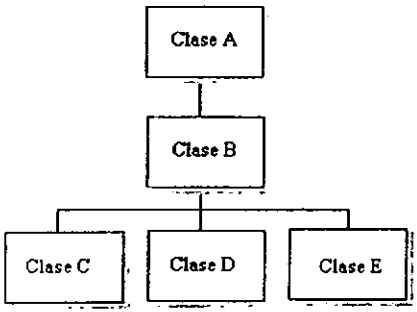
El agrupar en un entorno con límites bien definidos distintos elementos, se denomina encapsulación. En Java la abstracción y encapsulación la vemos representada por la clase, es donde se encierra y amalgama de forma clara tanto los datos de que constan los objetos, como los procedimientos que permiten manipularlos.



2.3.2 HERENCIA

La herencia permite la creación de objetos a partir de otros ya definidos. Se aplica sobre las clases, de alguna forma las clases pueden tener "descendencia", por lo tanto, heredaran algunas características de la clase "padre" y además añade rasgos propios.

Al disponer las clases con un formato de árbol genealógico, tendremos lo que se denomina una estructura jerarquizada de clases. La herencia tiene un efecto directo sobre la forma en que se diseñan las clases en Java.



Jerarquía de clase

Todas las clases están organizadas dentro de una jerarquía estricta. Cada clase tiene una superclase (la clase superior en la jerarquía), y puede tener una o más subclases. Las clases inferiores en la jerarquía heredan de las clases más altas. Las subclases heredan todos los métodos y variables de las superclases. La forma de heredar en Java se denomina Herencia Sencilla, no hay herencia múltiple por que cada clase solo puede tener una superclase, aunque pueda tener múltiples subclases.

Si por algún motivo es omitida la superclase, ésta tendrá en forma implícita la clase *Object* como superclase inmediata, por que es la raíz de la jerarquía de clases de Java. Así, todas son subclases directas o indirectas de la clase *Object*

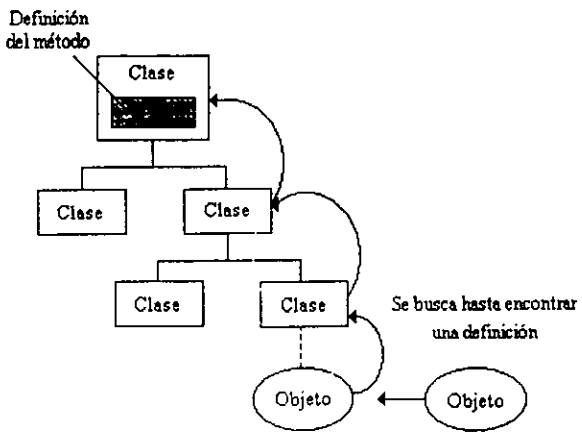
Cuando se crea una variable de instancia de una clase, se obtiene un "lugar" para cada variable definida en la clase actual , y para cada variable definida en todas sus superclases. así todas las clases se combinan para formar una plantilla para el objeto actual, y después cada objeto llena el lugar con la información adecuada para su situación.

2.3.3 POLIMORFISMO

Permite acceder a un variado rango de funciones distintas, a través del mismo identificador, es decir, un mismo identificador puede tener distintas formas de comportamiento. Esto es permitido en Java.

Cuando una subclase define un método con la misma identificación (nombre, numero y tipo de argumentos) de uno ya existente, el método que se ejecuta es el primero que encuentra, se empieza de abajo hacia arriba en la jerarquía. Así se pueden definir métodos con el mismo nombre, y se le denomina *sobreponer métodos*.

Los nuevos objetos tienen acceso a todos los nombres de métodos de su clase y sus superclases, pero las definiciones de método se eligen en forma dinámica cuando se llama a un método. Si se llama a un método de un objeto en particular, Java primero revisa en la clase del objeto al buscar la definición de ese método. Si no esta definida la clase del objeto, busca en la superclase de la clase , y así en lo sucesivo hacia arriba de la cadena, hasta que encuentra la definición del método.



Localización de métodos

2.4 PROGRAMACIÓN EN JAVA

Cuando se escribe un programa Java, se diseña y construye un conjunto de clases. Después, cuando se ejecuta el programa, las instancias de clases se crean y se descartan como sea necesario. La tarea como programador es crear el conjunto de clases adecuado para lograr lo que el programa necesita cumplir. Java es un lenguaje sensitivo, donde la indentación de cada parte de la clase no es importante para el compilador, sin embargo utilizarla hace la clase fácil de leer.

El lenguaje incluye una biblioteca (conjunto de clases) agrupados en paquetes, que implementa mucho de comportamiento básico que se necesita, no sólo para funciones matemáticas, arreglos, cadenas, etc., si no también para gráficos y comportamiento en red.

2.4.1 COMENTARIOS

Existen tres tipos de comentarios.

```
//      Comentarios para una sola línea

/*      comentarios
        de una
        o más líneas
*/

/**comentario de documentación, de una o mas lineas
*/
```

Los dos primeros tipos de comentarios son los que todo programador conoce y se utilizan del mismo modo . Los comentarios de documentación, colocados inmediatamente antes de una declaración (de variable o función), indican que ese comentario ha de ser colocado en la documentación que se genera automáticamente cuando se utiliza la herramienta proporcionada en el JDK, javadoc. Dichos comentarios sirven como descripción del elemento declarado permitiendo generar una documentación de nuestras clases escrita al mismo tiempo que se compila y genera el código.

2.4.2 PALABRAS RESERVADAS

abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe
byvalue	else	import	protected	throw
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
const	float	native	super	while

Además, el lenguaje se reserva unas cuantas palabras más, pero que hasta ahora no tienen un cometido específico. Son:

cast	future	generic	inner
operator	outer	rest	var

2.4.3 VARIABLES

Toda variable debe estar declarada dentro de una clase o un método. Tienen un nombre tipo y valor, antes de utilizarla se debe declarar y de ahí puede asignarsele un valor. Aquí simplemente no existen las variables globales.

Java utiliza cinco tipos de elementos: enteros, reales en coma flotante, booleanos, caracteres y cadenas, que se pueden poner en cualquier lugar del código fuente de Java. Cada uno de estos literales tiene un tipo correspondiente asociado con él.

Enteros:

byte	8 bits	complemento a dos
short	16 bits	complemento a dos
int	32 bits	complemento a dos
long	64 bits	complemento a dos

Por ejemplo: int x=21

Reales en coma flotante:

float	32 bits	IEEE 754
double	64 bits	IEEE 754

Por ejemplo: float x= 3.14 ; double y=2e12 ;

Booleanos:

boolean

Por ejemplo : boolean encendido = true ; boolean apagado = false ;

Cadenas:

String

Por ejemplo : String nombre="Esto es una cadena literal" ;

2.4.4 OPERADORES

Los operadores son símbolos especiales que, por lo común, se utilizan en expresiones. Los operadores de Java son muy parecidos en estilo y funcionamiento a los de C. En la siguiente tabla aparecen los operadores que se utilizan en Java, por orden de precedencia:

```
.    []   ()
++   --
!    ~    instanceof
*    /    %
+    -
<<   >>   >>>
<    >    <=    >=    ==    !=
&    ^    |
&&   ||
? :
=    op=  (*=   /=   %=   +=   -=   etc.)  ,
```

Los operadores numéricos se comportan igual que en C:

int = int + int

Los operadores relacionales devuelven un valor booleano.

Para las cadenas, se pueden utilizar los operadores relacionales para comparaciones además de + y = para la concatenación:

String nombre = "nombre" + "Apellido";

2.4.5 SEPARADORES

Sólo hay un par de secuencias con otros caracteres que pueden aparecer en el código Java; son los separadores simples, que van a definir la forma y función del código. Los separadores admitidos en Java son:

( ) - paréntesis. Para contener listas de parámetros en la definición y llamada a métodos. También se utiliza para definir precedencia en expresiones, contener expresiones para control de flujo y rodear las conversiones de tipo.

```
y = (5+3) * 5;          Encendido( y );
```

{} - llaves. Para contener los valores de matrices inicializadas automáticamente. También se utiliza para definir un bloque de código, para clases, métodos y ámbitos locales.

```
int j[3] = { 5, 29, 56 };
```

[ ]- corchetes. Para declarar tipos matriz. También se utiliza cuando se hace referencia a valores de matriz.

```
x = j[1] + j[2];
```

; - punto y coma. Separa sentencias.

```
y = 3+5;        j = 5+3;
```

, - coma. Separa identificadores consecutivos en una declaración de variables. También se utiliza para encadenar sentencias dentro de una sentencia for.

```
String Color, Modelo;          for(i=0, j=0; j<100; i++, j+=3){.....}
```

. - punto. Para separar nombres de paquete de subpaquetes y clases. También se

utiliza para separar una variable o método de una variable de referencia.

```
Moto.Color = Rojo;
```

2.4.6 SENTENCIAS DE SALTO

if/else. Permite ejecutar diferentes partes del código evaluando una condición.

```
if( Boolean ) {sentencias; }
else {sentencias; }
```

switch. Agrupa pruebas y acciones en un solo enunciado, se comportan igual que en C

```
switch( expr1 ) {
  case expr2: sentencias; break;
  case expr3: sentencias; break;
  default:
    sentencias;
    break;
  }
```

2.4.7 SENTENCIAS DE BUCLE

Bucles for

```
for( expr1 inicio; expr2 condicional; expr3 incremento ) {
   sentencias; }
```

También soporta el operador coma (,) en los bucles for for(a=0,b=0; a < 7; a++,b+=2)

Bucles while

```
while( Boolean ) {sentencias; }
```

Bucles do/while

```
do { sentencias; }while( Boolean );
```

2.4.8 EXCEPCIONES

Java cuenta con manejo de excepciones (intenta-atrapa), para posibles errores que puedan ocurrir, las excepciones son de la forma try-catch, y su sintaxis es la siguiente:

try {sentencias; }catch(Exepción) { sentencias; }

2.4.9 CREACIÓN DE OBJETOS

Un objeto en Java se crea por medio de una expresión dinámica, de la siguiente forma:

Nombre _clase objeto = *new* Nombre_clase(parámetros)

Ejemplo : Vehiculos Moto1 = new Vehiculos( );

La creación de un objeto, conlleva lo siguiente:

- Una asignación de memoria para el nuevo objeto, asociando valores iniciales a las variables del objeto.
- Invoca el constructor apropiado de la clase con los parámetros correspondientes, dependiendo de los parámetros se llamará al constructor adecuado.
- Regresa como resultado la referencia al objeto creado.

2.4.10 AGRUPACIÓN DE CLASES

Las clases en Java no andan sueltas, se agrupan en paquetes, estos son una manera de agrupar clases e interfaces relacionadas, se asocian con la palabra reservada *package*. Cada clase e interfaz se encuentra contenida en algún paquete, el cual tiene una nomenclatura separada por puntos, y por convención se indica como primer elemento la institución que genera el paquete. Esto permite que las clases estén disponibles sólo cuando

se necesitan , y eliminan conflictos entre los nombres de clase, en diferentes grupos de clase.

2.4.11 INTERFACES

Una interfaz es una colección de nombres de métodos sin definiciones reales que indican que una clase tiene un conjunto de comportamientos, además de los que la clase hereda de sus superclases.

Aunque una clase solo puede tener una superclase, esa clase puede implantar cualquier número de interfaces utilizando la palabra reservada implements para indicar el concepto. Al implantar una interfaz, se proporciona implantación de métodos definidos por la interfaz. Todas las clases que se utilizan en aplicaciones Java para internet serán derivadas de la clase Applet y tendrán como interfaz Runnable.

2.4. 12 PAQUETES

Para poder utilizar el código de algún paquete se requiere importarlo, utilizando la palabra reservada *import*. Por ejemplo, import java.applet.Applet;

Así, Java proporciona una serie de paquetes que incluyen ventanas, utilidades, un sistema de entrada/salida general, herramientas y comunicaciones. Algunos de los paquetes más importantes son:

java.applet

Este paquete contiene clases diseñadas para usar applets. Hay una clase Applet y tres interfaces: AppletContext, AppletStub y AudioClip.

java.awt

El paquete Abstract Windowing Toolkit (awt) contiene clases para generar componentes GUI (Interfaz Gráfica de Usuario). Incluye las clases Button, Checkbox, Choice, Component, Graphics, Menu, Panel, TextArea y TextField.

java.io

El paquete de entrada/salida contiene las clases de acceso a ficheros: FileInputStream y FileOutputStream.

java.lang

Este paquete incluye las clases del lenguaje Java propiamente dicho: Object, Thread, Exception, System, Integer, Float, Math, String, etc.

java.net

Este paquete da soporte a las conexiones del protocolo TCP/IP y, ademas, incluye las clases Socket, URL y URLConnection.

java.util

Este paquete es una miscelanea de clases útiles para muchas cosas en programación. Se incluyen, entre otras, Date (fecha), Dictionary (diccionario), Random (números aleatorios) y Stack (pila FIFO).

2.5 APPLETS

Java es un lenguaje de programación que puede emplearse para crear todo tipo de aplicaciones posibles, como al utilizar un lenguaje de programación tradicional. Los programas Java comprenden dos grupos principales:

- Applets

  Programas Java, que se cargan en el Web y se ejecutan por medio de un visualizador (con un run-time integrado) en la máquina cliente.

- Aplicaciones

  Programas individuales que se ejecutan al utilizar solo el intérprete de Java.

En general, un programa Java puede ser una aplicación, un applet o ambos, dependiendo de la manera en que se escriba y las capacidades que utilice el programa.

Los applets se ejecutan desde un visualizador Web, que deberá tener como característica, el run-time de Java integrado. Así, una referencia a un applet se incrusta en una página web al emplear una etiqueta HTML especial. Los applets tienen como ventaja la estructura que les ofrece el visualizador : Un contexto de eventos, gráficos e interfaz de usuario, además que liberan al servidor (donde están almacenados) de carga, pues para trabajar, utilizan los recursos de la máquina cliente donde son cargados. Sin embargo, esta ventaja esta obstaculizada por algunas restricciones impuestas como seguridad en su ejecución.

2.5.1 RESTRICCIONES

Las restricciones son necesarias para prevenir que cause daños al sistema o rupturas en la seguridad. Las restricciones en la ejecución de un applet, son las siguientes:

- No pueden leer o escribir en el sistema de archivos del lector, a excepción de directorios específicos, los cuales se definen por el usuario mediante una lista de control de acceso que, predeterminadamente esta vacía. Algunos visualizadores no permiten leer o escribir al sistema de archivos.
- No pueden ejecutar programas en el sistema del lector. Para los sistemas Unix, esto incluye atacar un proceso.

Así pues, las ventajas que los applets tienen sobre las aplicaciones (liberación de espacio en disco, estructura gráfica, etc.), está obstaculizada por las restricciones impuestas en su ejecución.

2.5.2 CREACIÓN Y CICLO DE VIDA DE UN APPLET

Para crear un applet, se debe definir una subclase de la clase Applet. Esta clase permite funcionar al applet dentro del visualizador y toma ventaja de las capacidades de AWT (Abstract Windowing Toolkit) para incluir elementos UI (User Interface), manejar elementos de ratón, palabras clave, así como dibujar en la pantalla. La clase principal del applet siempre tendrá una identificación como está:

Public class **Nombre_clase** extends java.applet.Applet{.....

Los applets tienen un ciclo de vida definido por cuatro métodos; **init( )**, **start( )**, stop( ), destroy( ).

Inicialización: Se lleva a cabo cuando se carga el applet por medio del método init( ), que inicializa los recursos que va a solicitar el applet, y los elementos que pueda contener tales como botones, cajas, etc. Este método se realiza una sola vez, esto es, se ejecuta cada vez que el applet se carga.

Arranque : Es el siguiente paso después de que se inicializa el applet. Comienza la ejecución del applet por medio del método start( ), lo cual ocurre si fue detenido con anterioridad, o cuando se vuelve a visitar la página que contiene al applet. Esto ocurre varias veces durante el ciclo de vida de un applet, mientras que la inicialización ocurre solo una vez. No es necesario llamarlo directamente, es llamado cuando el documento HTML es visitado.

Detención : La detención y el arranque van de la mano. La detención ocurre cuando se abandona la página que contiene un applet en ejecución, o cuando se llama al método stop( ), para detener la ejecución. Este método llama antes al método destroy( ). No se requiere llamarlo directamente.

Destrucción : La destrucción permite desalojar recursos solicitados por el applet. Se sobrepone el método destroy( ) cuando se tienen recursos específicos que necesiten liberarse.

Por ejemplo :

```
import java.awt.*;
import java.applet.*;
 public class Botones extends Applet{
   public void init(){
     setLayout(new FlowLayout(FlowLayout.RIGHT));
    setLayout(new GridLayout(3,3));
    add(new Button("Boton1"));
    add(new Button("Boton2"));
   }
}
```

2.5.3 ETIQUETA <APPLET>

Como se menciono anteriormente, un applet se incrusta en una página web al emplear una etiqueta HTML especial llamada Applet, que es una extensión especial de HTML. La directiva se abre con la etiqueta <APPLET> y se cierra con </APPLET>. Esta etiqueta incluye varios atributos que ayudan a integrar mejor el applet en el diseño general de la página Web.

El atributo CODE indica el nombre del archivo de clase que contiene al applet, incluida la extensión class. En este caso, el archivo de clase debe encontrarse en el mismo directorio que el archivo HTML.

WIDTH y HEIGHT son necesarios, se utilizan para indicar el cuadro de fondo para el applet, es decir, que tan grande será el cuadro para el applet en la página Web. Se debe

asegurar la configuración correcta de estos elementos al tamaño adecuado del applet, por que se podría perder parte de este.

```
<Applet  Code="Hola.class"  Width= 100  Height= 20>
Su navegador Web no soporta Java...
</Applet>
```

El texto entre las etiquetas de apertura y cierre, se desplegará por visualizadores que no entiendan la etiqueta. Es importante incluir texto alterno para que quien tenga un navegador que no soporte Java vea más que una línea en blanco. Por ejemplo, se puede incluir la siguiente cadena: Su navegador Web no soporta java.....

El atributo ALIGN define la alineación que tendrá el applet en la página. Este atributo toma uno de nueve valores:

LEFT, RIGHT: El applet se coloca en los márgenes, izquierdo o derecho respectivamente, de la página, y todo el texto va en el espacio a la izquierda o derecha.

TEXTTOP: Alinea la parte superior del applet con la parte superior del texto en la línea.

MIDDLE: Alinea el centro del applet con el centro de la línea base del texto.

ABSMIDDLE: Alinea el centro del applet con el centro del elemento más grande en la línea.

BASELINE Y BOTTOM: Alinea la parte inferior del applet con la línea base del texto.

ABSBOTTOM: Alinea la parte inferior del applet con el elemento más bajo en la línea.

Los atributos HSPACE Y VSPACE se emplean para configurar la cantidad de espacio, en pixeles, entre un applet y su texto circundante. HSPACE controla el espacio horizontal y VSPACE controla el espacio vertical.

CODEBASE, se utiliza cuando se guardan los archivos de clase en un directorio diferente al de los archivos HTML.

En aplicaciones se pueden pasar argumentos a la rutina main( ) por medio de la línea de comandos. Después de analizar esos argumentos dentro del cuerpo de la clase, la aplicación actuará de acuerdo con los argumentos que se le dieron.

```
java Hola.class "Pepe"
```

Donde obtenemos en la pantalla como resultado: ! Hola, bonito nombre Pepe ¡

Los applets no cuentan con una línea de comando, tienen una entrada distinta que contiene la etiqueta <Applet>. Incluyen dos partes: un nombre y un valor que determina ese parámetro en particular. Por ejemplo se puede indicar el color del texto en un applet al utilizar el parámetro de nombre=color y su valor=azul. Esto se indica por medio de la etiqueta <PARAM>, la cual tiene dos atributos para el nombre y su valor llamados NAME y VALUE.

```
<Applet Code="Hola.class" Width= 100 Height= 20>
<Param Name=Color Value=Azul>
<Param Name=Nombre Value=Pepe>
Su navegador Web no soporta Java...
</Applet>
```

2.5.4 VISUALIZADOR WEB

El procedimiento de ejecución que realiza el navegador Web al encontrar la marca <Applet> en el código HTML es el siguiente:

- Carga el código de bytes asociado al applet efectuando la transmisión de este, desde el servidor Web hasta el cliente.
- Se crea una instancia de la subclase applet.
- Se llama al método init ( ) para efectuar el proceso de inicialización.
- Se llama al método start ( ) para comenzar la ejecución de applet.

En el momento en que un applet se encuentra corriendo, el visor Web se encarga de manipular otras operaciones asociadas al comportamiento de este. Cuando se minimiza la ventana del visor, un applet detiene su ejecución mediante la operación stop ( ), con objeto

de no consumir tiempo de CPU cuando no se muestra la página que lo contiene. Lo anterior se debe a que cada uno de esos métodos los ejecuta de forma automática la Maquina Virtual Java en el visor Web.

Capitulo 3

Características del Problema

En este capítulo se explica de manera rápida un problema que se usará como base para ilustrar de una manera más efectiva la metodología de desarrollo de applets.

3.1 MARCO HISTÓRICO

3.1.1 HISTORIA

La Universidad Nacional Autónoma de México, por acuerdo del rector en 1986, creó la comisión de educación continua de la UNAM, presidida por el secretario general y desde entonces se han desarrollado actividades académicas, que incluyeron la realización del primer seminario de educación continua en 1988.

Por medio del programa de educación continua la coordinación de programas académicos impulsa el fortalecimiento de las actividades académicas en todas las materias, propicia la vinculación entre las dependencias universitarias y con la sociedad, desarrollando las modalidades educativas y las estrategias más diversas, que permiten ofrecer actividades de actualización en forma presencial en las instalaciones del campus universitario, en instalaciones externas y con Programas de Educación a distancia.

3.1.2 ORÍGENES

Redec esta integrada por representantes designados por los directores de facultades, escuelas, institutos, centros, programas y todas las dependencias universitarias interesadas en ofrecer programas de educación continua.

Inicio sus trabajos en noviembre de 1995, bajo la conducción del secretario general, con el apoyo de la coordinación de programas académicos.

Redec realiza reuniones periódicas de enlace para fortalecer los canales para intercambiar experiencias, propiciar la colaboración estratégica entre las dependencias universitarias. Ha realizado trabajos para aportar diagnósticos, alternativas de vinculación, lineamientos operativos, criterios de calidad, bases normativas y fundamento teórico para las actividades de Educación Continua, acordes con necesidades de actualización de conocimientos, adquisición de habilidades, certificación de competencias profesionales y con nuevas relaciones de carácter interdisciplinario que exige la sociedad actual.

3.2 PLANTEAMIENTO DEL PROBLEMA

3.2.1 NECESIDADES

El interés de redec por mantener una relación permanente con todos los sectores de la sociedad, la ha llevado a preparar la Pagina de Redec UNAM, en Internet y él catalogo de educación continua redec, en los cuales se podrán encontrar los programas, cursos y otras modalidades que ofrece cada centro integrante de la red, con la descripción y la exposición de su potencial educativo y las posibilidades de ampliar la vinculación entre la sociedad y la universidad, para la búsqueda común de soluciones a las necesidades de actualización del conocimiento y la construcción de alternativas de aplicación inmediata para enfrentar retos actuales y futuros.

Redec es una oferta educativa de actualización, formación de personas y organizaciones de todos los grupos sociales, requiere la atención permanente a la calidad de los productos educativos que la conforman. El servicio que brinda es a través de

acciones amplias y variadas, por lo cual su evaluación es muy compleja y entre otros aspectos es importante destacar que mal del 20% de los asistentes reciben beca completa. Redec ha estimado de primera importancia la participación amplia de los miembros de la comunidad en el gran potencial de servicios educativos que puede proporcionar la UNAM, en forma presencial y de educación a distancia y las múltiples áreas temáticas abordadas por expertos que producen el conocimiento de punta en ciencias y humanidades en México y el mundo, para beneficio de amplios sectores de la sociedad.

Una convicción es que la reunión de fortalezas en la educación continua, ampliara las posibilidades de ofrecer programas de calidad, proporcionará beneficios a todos los participantes y contribuirá a la discusión de la imagen y la riqueza de la institución en las comunidades especificas a las que beneficien sus programas y en toda la sociedad mexicana. Las formas de realización de acciones, se puede expresar en programas específicos entre las instancias interesadas, o en convenios de alcance mayor, con formas para evaluar y retroalimentar lo realizado par enriquecer y diversificar nuevas acciones que permitan aprovechar oportunidades para vencer nuevos retos.

3.2.2 MÉTODO ACTUAL DE PREINSCRIPCION

Por el momento la manera de pre-inscribirse a un curso es la siguiente:

- Consultar listas de cursos: Se elige el curso a tomar.
- Presentarse por una forma de preinscripción: Que incluye todos los datos del candidato y la cual deberá ser llenada a máquina.
- Consultar listas de candidatos aceptados: Se envían los datos a las dependencias que impartirán el curso. Estas listas son publicadas en las ventanillas.
- Checar si el curso elegido no se cancelo.
- Presentarse, por lo menos 2 días antes, para inscripción y pago.

Como se puede observar, en la actual metodología de inscripción, es necesario presentarse más de una vez en las oficinas correspondientes para poder obtener la inscripción a algún curso. Estos trámites, para personas que trabajen, pueden llegar a ser demasiado engorrosos, por lo tanto, esta pagina dará gran flexibilidad a la inscripción y permitirá un manejo más fácil y rápido de la información que se maneja.

3.2.3 DISEÑO DE LA PÁGINA

La página de redec en Internet, tendrá un diseño que permitirá la consulta del usuario por nombre de curso, clave y fecha. El sistema permitirá obtener un calculo de la cantidad total que se deberá pagar por la inscripción a uno o más cursos, asimismo, se podrá hacer una preinscripción. Contará con ligas a otras páginas, tanto de instituciones nacionales como extranjeras. Cada centro podrá actualizar directa y permanentemente su información para contar en todo momento con una página dinámica.

3.2.4 FORMATO DE PRESENTACIÓN

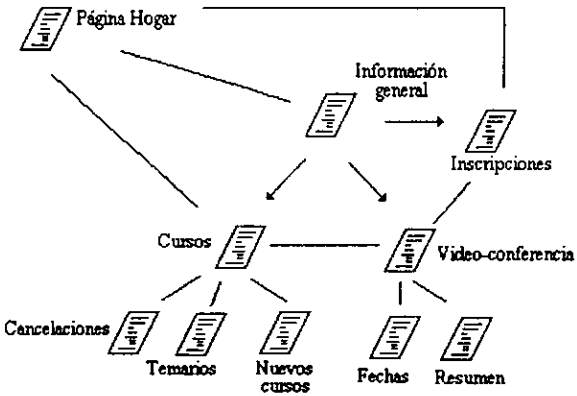
Descripción del formato para presentar la información de los cursos, para su publicación en la página en Internet y la presentación impresa en los catálogos.

Al inicio de la página habrá una presentación general de los objetivos que se persiguen. Posteriormente se sucederán tantas páginas como centros de educación continua estén interesados en formar parte de la página central.

Los cursos se presentarán por cada centro de educación, con una imagen de su elección que considere representativa y un texto con la descripción general de temas,

perfil de ponentes, recursos, modalidades y servicios educativos en general que ese centro ofrece.

Estas páginas estarán distribuido como se muestra en la figura, cada una con un contenido particular.



La página principal, debe contener:

1. El logotipo oficial de REDEC.
2. La bienvenida para los visitantes.
3. Ligas a cada uno de los principales servicios.

En la página de Información General, se debe proporcionar:

1. Historia y origen de la organización
2. Objetivos principales
3. Datos de localización: Teléfono, dirección y horarios.

4. Ligas a los otros servicios.

En la página de Video-Conferencia, se dispondrá de:

1. Fechas de futuras conferencias, locales y remotas.
2. Resumen de los temas de las conferencias de la semana.

En la página de cursos, se deberá disponer de:

1. Datos generales de los cursos.
2. Temario de cada uno de los cursos.
3. Cursos de cancelados.

La página de inscripciones es la que explicare todo a detalle, así pues, debe cubrir los siguientes puntos:

1. Opción de búsqueda de cursos
   a) Por nombre
   b) Por fecha
   c) Clave

Como resultado, deberá mostrar en la pantalla:

Tipo	Especifica la actividad de que se trata: DIPLOMADO, CURSO, etc.
Título:	Nombre de la actividad
Clave:	Letras y dígitos
Expositor:	Nombre del ponente
Fecha:	Se expresa con cuatro dígitos, dos del año y dos del mes.
Sede:	Lugar o lugares donde se imparte.

Cada actividad tiene una clave con 4 letras, y 5 dígitos.

A. 4 Caracteres para identificar el centro de educación donde será impartido el curso
B. 2 Dígitos para identificar el año de realización de la actividad
C. 3 Dígitos para una identificación distintiva de cada actividad

Para el grupo C los dígitos se utilizarán con la siguiente convención:

101 al 199	Diplomados de cada centro de educación.
161 al 169	Diplomados con instituciones nacionales.
171 al 179	Diplomados con instituciones extranjeras.
181 al 189	Diplomados a distancia.
191 al 199	Diplomados con otras características distintivas especificadas explícitamente.
201 al 299	Cursos (incluye seminarios, talleres y otros)
301 al 599	Cursos con otros centro de educación continua de la misma institución.
601 al 699	Cursos con otras instituciones nacionales.
701 al 799	Cursos con otras instituciones extranjeras.
801 al 899	Cursos a distancia y otras características distintivas.

Un ejemplo simple: Curso: Análisis de Políticas Públicas para Asuntos de Gobierno
Clave: ENAC.96.701

Ejemplo de la presentación de actividades en la página en Internet y en el catálogo.

FACULTAD DE ARQUITECTURA: DIPLOMADOS Y CURSOS DE EDUCACIÓN CONTINUA

Tipo	DIPLOMADO
Título:	MANEJO DE COMPUTADORAS COMPATIBLES PC
Clave:	FARQ.96.101
Expositor:	Fis. Alejandro Mendez
Fecha:	9608(agosto-febrero 97)
Horarios:	Martes y jueves de 17:00 a 21:00 horas.
Sede:	Aulas de la Facultad de Arquitectura

2. Formato de inscripción

a) Debe contener todos los detalles del candidato.

Nombre
Dirección – teléfono
E-mail

3. Cálculo de precio a pagar

Dependiendo del instituto o escuela de origen del candidato, este tendrá derecho a descuentos, por este motivo, la página debe contar con una forma para cálculo de la cantidad total a pagar.

Todos y cada uno de los puntos que se desarrollarán en la página son importantes, en conjunto proporcionan interatividad con la página.

Capitulo 4

Metodología de Desarrollo de Applets

4.1 MARCO TEÓRICO

Es importante comprender y comunicar los requisitos de un sistema antes de programarlo, en nuestros días, se continúan buscando métodos para mejorar esta fase del proceso de desarrollo. Los métodos antiguos descansaban en descripciones textuales realizadas sobre papel. Estos métodos conducían no solo a la ambigüedad sino que también dificultaban las modificaciones y eran incapaces de apoyar el desarrollo de sistemas muy grandes.

4.1.1 ESTILOS DE PROGRAMACIÓN

El análisis estructurado, basado principalmente en descomponer componentes funcionales de un sistema, fue desarrollado en los años 60 e introdujo un método definitivo y más manejable para el análisis de sistemas. El Análisis de sistemas Orientado a Objetos es un nuevo método que realza la definición de las características y el comportamiento dentro de un sistema de objetos.

En términos generales los expertos sugieren que en general existen cuatro tipos de estilos de programación:

- Orientado a procedimientos — Algoritmos estructurados
- Orientado a objetos — Clases y Objetos
- Orientado a lógica — Expresiones obtenidas en cálculo de predicados
- Orientado a reglas — Reglas if-then

No existe un solo estilo de programación que sea la mejor para todo tipo de aplicaciones. Cada uno de estos estilos de programación esta basado sobre su propio marco de trabajo, y cada uno requiere su propia manera de pensar el problema.

4.1.2 INICIO DE LOS MÉTODOS ORIENTADOS A OBJETOS

El modelado de información, popularizado por Peter Chenen a década de los ochenta, es el precursor mas próximo del análisis y diseño orientados a objetos. Aquí el resultado final del análisis de modelado de información es un diagrama entidad-relacion, aunque este método esta íntimamente relacionado con el análisis orientado a objetos, no existe encapsulacion de datos. El método de modelado de información no permite la herencia ni el paso de mensajes.

El primer material publicado sobre el análisis orientado a objetos provino de Sally Shlaer y Stephen Mellor (1988). Su método comienza por definir objetos y atributos. A continuación se definen los ciclos de vida de los objetos en los modelos de estado. Para capturar los sucesos que actúan sobre los objetos. El ultimo paso es la definición de procesos basada en los objetos y sus ciclos de vida. A diferencia de los métodos de descomposición funcional y de suceso-respuesta, el método de orientación a objetos se traduce en un mínimo código derivado de los datos, que permanece estable incluso ante el cambio de requisitos.

En un sistema orientado a objetos no recae el énfasis en la transformaciones de entradas en salidas, sino en el contenido de entidades, en los objetos.

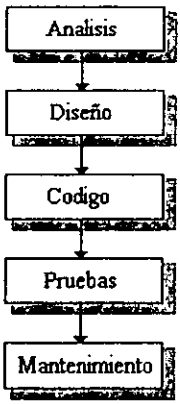
Se trata de agrupar métodos cuando estos funcionan sobre una misma abstracción de datos. Es el paso de mensajes entre objetos lo que determina la secuencia de funcionamiento.

Enfatizar sobre la frase "Método Orientado a objetos" se refiere a cosas similares como "orientación a objetos" y "tecnología de objetos", por si mismas. En particular la frase

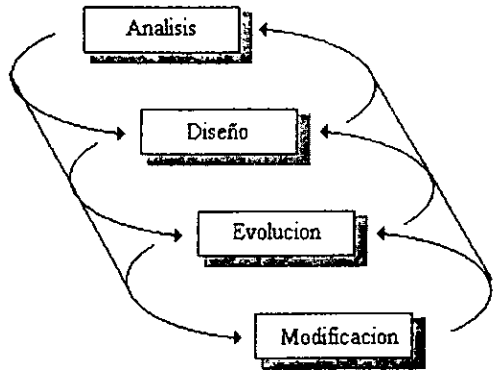
se refiere a programación orientada a objetos, diseño y análisis, hacia una misma filosofía de desarrollo de sistemas basados sobre una metáfora poderosa.

4.1.3 CICLO DE VIDA

Un método siempre consiste de una notación (representación gráfica) y de un proceso. Una aplicación orientada a objetos no puede ser desarrollada usando el tradicional ciclo de vida del software.



Es mejor adoptar un proceso de desarrollo cíclico. Esto significa desarrollar la aplicación en pequeñas piezas. Se inicia una aplicación con un mínimo de funcionalidad y gradualmente se agregan todas las características necesarias.



Aquí observamos que el diseño no es distinto del tradicional, sin embargo, estos pasos están retroalimentados unos con otros y no es un flujo rígido de arriba hacia abajo solamente. Por que tenemos la aplicación trabajando, se tiene una continua retroalimentacion y se puede eventualmente adaptar el diseño. La metodología OO asume que durante el ciclo de vida de la aplicación, las fases de análisis y diseño son atravesadas iterativamente e incrementalmente

4.2 MODELADO ORIENTADO A OBJETOS

4.2.1 OBSERVACIONES SOBRE LA CRISIS DEL SOFTWARE.

El término "crisis de software" indica la dificultad en el desarrollo de programas al controlar costos, tiempos y calidad. Desde siempre el hardware ha sido desarrollado con más velocidad.

En particular entre los muchos problemas que una fase o una pieza de software atraviesa (esto es, las varias fases que el software atraviesa desde su formulación hasta su

uso efectivo), los problemas de mantenimiento son los que se presentan frecuentemente y mas severos. Entre las cinco fases que generalmente se distinguen (análisis, diseño, implementación, depuramiento y mantenimiento) el mantenimiento estuvo considerado de secundaria importancia. Hoy, muchos investigadores están convencidos que el mantenimiento absorbe más de la mitad del costo global y recursos en el desarrollo del software.

Se descubrió en muchos estudios hechos durante los últimos años, que una vez que la primera versión está funcionando, se necesito mucho tiempo antes que está pueda ser considerada como liberada.

4.2.2 DESCRIPCIÓN DE LA REALIDAD A TRAVÉS DE ABSTRACCIONES

El arte de programación consiste en describir en una computadora, de un modo inteligente por medio de un lenguaje de programación, un fenómeno, una acción, un comportamiento o una idea.

En otras palabras, se necesita explicar como la computadora trabaja para ejecutar cálculos muy complejos o muy sencillos , que necesitan ser ejecutados para reproducir sobre la computadora un fenómeno fisico del cual se conoce el modelo matemático, simulando el comportamiento del sistema.

Se puede utilizar la computadora para simular el comportamiento de otra máquina. Por ejemplo, esta puede ser usada como una máquina de escribir, una herramienta de dibujo, una calculadora, una agenda o como un simple directorio telefónico.

La computadora necesita comportarse de cierta manera hasta lograr un comportamiento lo más parecido a el sistema que se desea modelar. Por lo tanto, los programadores necesitan dar a la computadora una descripción abstracta de este comportamiento.

Las abstracciones ofrecidas por lenguajes de programación están divididas en dos categorías las que pertenecen a los datos y las que pertenecen a las estructuras de control.

4.3 METODOLOGÍA ORIENTADA A OBJETOS

4.3.1 UNA METODOLOGÍA ORIENTADA A OBJETOS

En metodologías de diseño orientado a objetos actualmente no hay un método único e universal, pero sin embargo existe una familia de metodologías que pueden ser totalmente diferentes una de otra, sin embargo, todas ellas comparten un balance entre datos y funciones.

Las diferencias existen por muchas razones, pero la fundamental es que han sido concebidas para diferentes lenguajes de programación orientados a objetos. Se considera, que la más completa y formal es la metodología orientada a objetos es la de Grady Booch[1].

4.3.2 PROPIEDADES GENERALES DE LA METODOLOGÍA

El resultado de un diseño orientado a objetos es una jerarquía de clases. cada clase es un módulo separado con sus propias estructuras de control y datos. Se puede ver la extensión del problema de forma más natural y realista como un conjunto de objetos y métodos asociados. Los elementos iniciales de un método orientado a objetos, son los propios objetos. posteriormente, a medida que se identifican aspectos comunes, los objetos se van agrupando en clases que a su vez serán subclases de clases más abstractas. El nivel superior de abstracción se conoce como marco estructural. Un marco estructural es un conjunto de clases que expresan un diseño para una familia de aplicaciones relacionadas

[1] En determinado momento una metodología se podrá intercalar con otra ya que:
Las metodología no deben estar peleadas, si esto ocurre no sirven..... Grady Booch

entre sí. Los métodos de objetos definen un conjunto de módulos de comunicación independientes con visibilidad limitada entre ellos.

4.3.3 PASOS DE DESARROLLO

1. **Definición del dominio del problema.**

   Es la primera fase de en donde se unen los usuarios y los desarrolladores de un sistema, ellos unidos, dibujan el dominio del problema con un lenguaje común. "El propósito del análisis es proveer una descripción del problema. La descripción debe ser completa, consistente, leible y revisable por las diversas partes interesadas, los más cercano a la realidad. El producto del análisis es después utilizado para articular las funciones del sistema. Este punto fue ampliamente detallado en el capítulo anterior.

2. **Identificación de clases y objetos.**

   En términos generales, los límites entre el análisis y el diseño son muchas veces difusos. En el análisis, vemos el modelo del mundo identificando las clases y objetos del vocabulario del dominio del problema; en diseño orientado a objetos, se inventan las abstracciones y mecanismos que proveen el comportamiento que el modelo requiere. Es absolutamente crítico separar las actividades de análisis y diseño.

   El diseño de un sistema orientado a objetos comienza con los objetos. Encontrarlos es quizá el principal desafío del diseño y el análisis orientado a objetos. Se han utilizado varios métodos para identificar objetos, incluyendo la inspección gramatical de documentos y la derivación a partir de diagramas de flujo de datos y relación de entidades.

Comenzaremos por una definición corta del problema.

Definición del problema:

Preparar la Página de Redec UNAM, en internet y el catalogo de educación continua Redec.

Ahora haremos una descripción de la solución del problema, calificando los sustantivos.

Descripción de la solución:

La página de Redec en internet, tendrá un diseño que permitirá consultar cursos para el usuario por nombres, clave y fecha. El sistema de consulta permitirá al usuario obtener un calculo de la cantidad total que se deberá pagar por la inscripción a uno o más cursos, asimismo, se podrá hacer una preinscripción.

Una vez identificados, se sugiere una asociación de atributos con cada uno de ellos. Hay que observar que no todos los nombres terminarán por convertirse en objetos. Aunque este método gramatical es una buena forma de desarrollar una lista inicial de posibles clases, generará un cierto número de conceptos que no pertenecen al sistema a modelar y no tienen que introducirse al software, por todo esto, solo se toman las palabras clave:

Objeto	**Atributos**
Cursos	Se puede consultar.
	Se puede preinscribir.

Cálculo	Contiene uno o más cursos y se pueden sumar y/o restar.

Hay dos tipos de clases en el sistema:

- Las clases que se reflejan directamente en los conceptos del dominio de la aplicación, esto es, conceptos que son usados por usuarios finales para describir sus problemas y soluciones.

- Las clases que son "artefactos" de implementación, usadas por diseñadores y programadores para describir técnicas de implementación. Es importante observar que los detalles de una aplicación son diferentes de una persona a otra.

3. Análisis de objetos y su funcionalidad

En este punto, se caracteriza el comportamiento de un objeto o de una clase de objetos. Al final de estos pasos, se puede tiene una lista de operaciones de cada objeto.

El siguiente paso consiste en identificar los métodos posibles subrayando los verbos, de la forma siguiente:

> La página de Redec en internet, tendrá un diseño que permitirá consultar cursos para el usuario por temas, profesores, clave y fecha. El sistema de consulta permitirá al usuario obtener un calculo de la cantidad total que se deberá pagar por la inscripción a uno o más cursos, asimismo, se podrá hacer una preinscripción.

Ahora se recogen los métodos con sus objetos correspondientes, de la siguiente forma:

Objeto	Métodos
Cursos	Consulta.
	Preinscripcion.
Cálculo	Sumar
	Restar

4. Escribir las interfaces de las clases

La definición de interfaces entre objetos es paso final del proceso. Una vez más se crea una descripción descrita :

Se realiza el sistema como dos clases, **Cursos** y **Cálculo.**

1. La clase **Cursos** tendrá una variable de instancia llamada Curso_id, y los métodos siguientes:

- **Consulta**
- **Preinscripcion.**

2. La clase **Cálculo** contiene una variable de instancia llamada Calculo_id, y los métodos siguientes:

- **Suma**
- **Resta.**

El método de Booch proporciona una estrategia sencilla para identificar objetos y métodos analizando una descripción escrita y asignando nombres a objetos y verbos a clases o métodos. Este método de nombre y verbo es un comienzo razonable, pero oculta parte dela complejidad de la definición de clases. Con frecuencia se necesitan clases para operaciones en la parte verbal del problema.

Este ejemplo consta sobre todo de verbos que describen el funcionamiento del sistema sobre una búsqueda, pero al consultar en tiempo real hay que identificar objetos independientes del sistema.

Aparte del método Booch, las metodologías para apoyar el análisis y el diseño orientado a objetos se encuentran en sus primeras etapas de desarrollo, y se dispone de pocas herramientas automatizadas. La mayoría de los primeros usuarios de lenguajes y herramientas orientados a objetos han dependido de combinaciones metodológicas basadas en papel y experiencias publicadas por expertos.

Capitulo 5

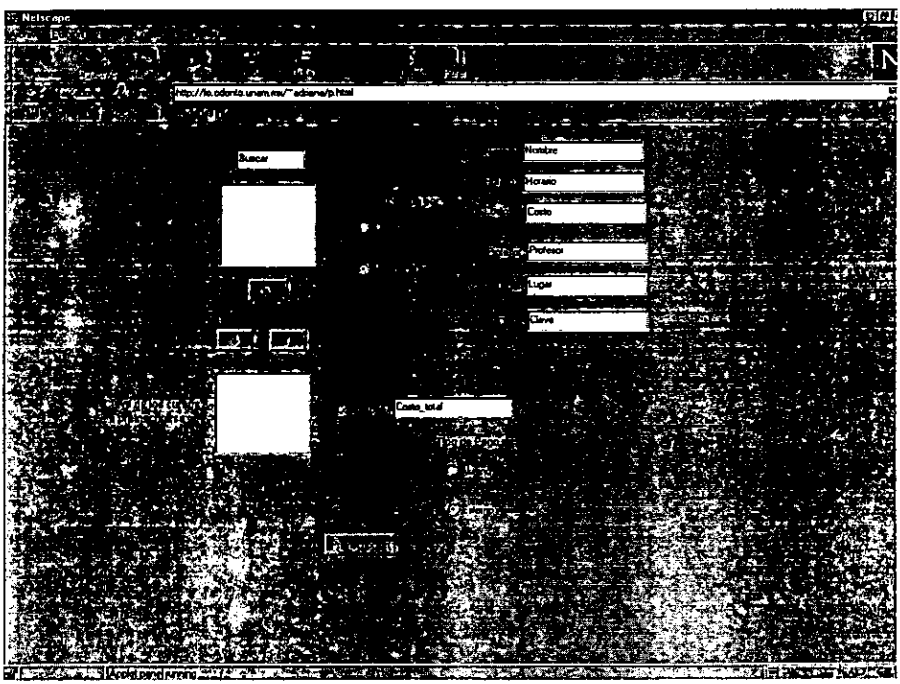
Construcción de Applets

Diversos autores coinciden en señalar que nuestra sociedad se encamina hacia una "Era de la Información" ya que, si bien las computadoras han estado presentes desde hacia varia décadas, la velocidad de los cambios en las últimas dos décadas ha brindado a los individuos y a las organizaciones capacidades que hasta hace poco no existían. Dentro de estos cambios tenemos la comprensión gráfica de la información.

Tomando en cuenta el problema analizado, debemos construir las clases necesarias y posteriormente colocar el applet obtenido dentro de una pagina web como veremos a continuación.

5.1 INTERFAZ DE USUARIO

Utilizaremos la AWT (Abstract Windowing Toolkit) para construir un sistema de interfaz gráfico, que nos permitira crear una estructura para el applet. Esta interfaz estará constituida como se muestra en la figura.



Debemos recordar, que el sistema de ventanas será diferente dependiendo de la plataforma en que se observe el programa, aunque la distribución es la misma.

```
public void init(){

add(palabra_buscar);
  add(lista);
  c.addItem("Nombre");
  c.addItem("Clave");
  add(c);

nombre=new creaEstructura("","Nombre",abc);
  add(nombre);
  hora=new creaEstructura("", "Hora ",abc);
  add(hora);
  costo =new creaEstructura("", "Costo ",abc);
  add(costo);
  profesor =new creaEstructura("", "Profesor ",abc);
  add(profesor);
  lugar =new creaEstructura("", "Lugar ",abc);
  add(lugar);
  clave =new creaEstructura("", "Clave",abc);
  add(clave);

suma=new Calculo(this,0.0,true);
  add(suma);
  resta=new Calculo(this,0.0,false);
  add(resta);
 add(claves_cursos);

add(new Label("Costo total de inscripcion"));
   add(new TextField(10) ;//para poner el costo total
```

```
        Button preinscripcion=new Button("preinscripcion");
        add(preinscripcion);

        actualiza(nombre);
    }
```

5.2 CLASE CALCULO

Esta clase nos permitirá realizar los cálculos totales que se deberán pagar para inscribirse al curso(s) desado(s), por medio del método suma. Asi, pues al descartar un curso de la lista se deberá hacer una operación de resta, por medio del método resta( ). El código necesario para lograrlo es el siguiente:

```
public class calculo( ){
float resultado;
    String titulo;
    double valor;
    boolean calculo_id;
    Cursos outerparent;

     Calculo(Cursos target,double Valor_operacion,boolean operacion){
      valor=Valor_operacion;
      calculo_id=operacion;
     this.outerparent=target;
     if(calculo_id)  titulo="Suma";
     else  titulo="Quita";

     add(new Button(titulo));
    }

    public boolean action(Event evt, Object arg){
```

```
      if(evt.target instanceof Button){
                System.out.println("Hola si paso");
                outerparent.update(this);
                return true;
        }
       else return false;
}//fin del handle


float  Suma(float x1, float y1){
 resultado = x1 + y1;
}
float  Resta(float x1, float y1){
 resultado = x1 - y1;
}
}
```

5.3 CLASE CURSO

Esta clase nos permitirá consultar los cursos existentes, con el método consulta(). También, podremos realizar una preinscripcion, método preinscripcion( ), evitándonos trámites engorrosos. El código es el siguiente:

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;


public class Cursos extends Applet implements Runnable{
  Choice c=new Choice();
  Calculo suma,resta;
  CheckboxGroup cbg=new CheckboxGroup();
  CheckboxGroup abc=new CheckboxGroup();;
  List lista=new List(10,false);
```

```
    TextField palabra_buscar=new TextField(40);
    Thread runner;
    creaEstructura nombre,hora,costo,profesor,lugar,clave;
    Frame window;
    List claves_cursos=new List(5,false);
  public void init(){
 add(palabra_buscar);
    add(lista);
    c.addItem("Nombre");
    c.addItem("Clave");
    add(c);
 nombre=new creaEstructura("","Nombre",abc);
    add(nombre);
    hora=new creaEstructura("", "Hora ",abc);
    add(hora);
    costo =new creaEstructura("", "Costo ",abc);
    add(costo);
    profesor =new creaEstructura("", "Profesor ",abc);
    add(profesor);
    lugar =new creaEstructura("", "Lugar ",abc);
    add(lugar);
    clave =new creaEstructura("", "Clave",abc);
    add(clave);

 suma=new Calculo(this,0.0,true);
    add(suma);
    resta=new Calculo(this,0.0,false);
    add(resta);
    add(claves_cursos);

 add(new Label("Costo total de inscripcion"));
     add(new TextField(10));
  Button preinscripcion=new Button("preinscripcion");
```

```
        add(preinscripcion);
        actualiza(nombre);
    }
    public void start(){
     if(runner==null){
      runner=new Thread(this);
      runner.start();
     }
    }
    public void stop(){
     if(runner!=null){
      runner.stop();
      runner=null;
     }
    }
    void Asignacion(String valor,int posicion,int aquien){
     if(aquien==0){
       nombre.Valor_caja[posicion]=valor;
nombre.Valor_caja[posicion]=nombre.Valor_caja[posicion].toUpperCase();
      }
     if(aquien==1){
       hora.Valor_caja[posicion]=valor;
hora.Valor_caja[posicion]=hora.Valor_caja[posicion].toUpperCase();
      }
     if(aquien==2){
       costo.Valor_caja[posicion]=valor;
costo.Valor_caja[posicion]=costo.Valor_caja[posicion].toUpperCase();
      }
     if(aquien==3){
       profesor.Valor_caja[posicion]=valor;
profesor.Valor_caja[posicion]=profesor.Valor_caja[posicion].toUpperCase()
;
```

```
        }
       if(aquien==4){
         lugar.Valor_caja[posicion]=valor;
lugar.Valor_caja[posicion]=lugar.Valor_caja[posicion].toUpperCase();
        }
       if(aquien==5){
         clave.Valor_caja[posicion]=valor;
clave.Valor_caja[posicion]=clave.Valor_caja[posicion].toUpperCase();
        }

     }



     public boolean handleEvent(Event evt){
     Character tecla;
     String t;
     String llave="";
      if(evt.target instanceof TextField){
        switch(evt.id){
          case Event.KEY_PRESS:
                  int a;
                  llave=palabra_buscar.getText();
                  Encuentra(llave);
                  a=lista.getSelectedIndex();
                  Muestra(a);
                  break;
               }
        }
     if(evt.target instanceof Choice){
        switch(evt.id){
          case Event.ACTION_EVENT:
```

```
                System.out.println("Existo choice");
                if(c.getSelectedItem()=="Nombre") actualiza(nombre);
                else actualiza(clave);
                break;
              }
        }
    if(evt.target instanceof List){
       switch(evt.id){
        case Event.LIST_SELECT:
               int a;
               a=lista.getSelectedIndex();
               Muestra(a);
               break;

       }
    }
      return super.handleEvent(evt);
    }
   void Encuentra(String en){
    int m=0,tem=0;
    boolean ban=false;
    en=en.toUpperCase();
    try{
       while((nombre.Valor_caja[m].startsWith(en)==false)&&(m<10)){
          tem=m+1;
          m++;
          ban=true;
          }
        if(ban) lista.select(tem);
        if(!ban) lista.select(0);
      }
      catch(NullPointerException e)';!
      }
```

```
 void Muestra(int sele){
    nombre.caja.setText(nombre.Valor_caja[sele]);
    hora.caja.setText(hora.Valor_caja[sele]);
    costo.caja.setText(costo.Valor_caja[sele]);
    profesor.caja.setText(profesor.Valor_caja[sele]);
    lugar.caja.setText(lugar.Valor_caja[sele]);
    clave.caja.setText(clave.Valor_caja[sele]);
 }
void update(Calculo in){
  if(in==suma){
     System.out.println("Soy suma");
     claves_cursos.addItem(clave.caja.getText());
     }
  else {
       System.out.println("Soy Resta");
       claves_cursos.delItem(claves_cursos.getSelectedIndex());
       }
   }

void actualiza(creaEstructura tem){
DataInputStream datos;
String linea;
StringBuffer buf=new StringBuffer();
String temp;
int j=0, posicion=0;
int k,l;
String asignacion;
try{
  System.out.println("Conexion abierta..");
  datos=new DataInputStream(new FileInputStream("datos.dat"));
  System.out.println("Leyendo datos...");

  for(int a=0;a<lista.countItems();a++){
```

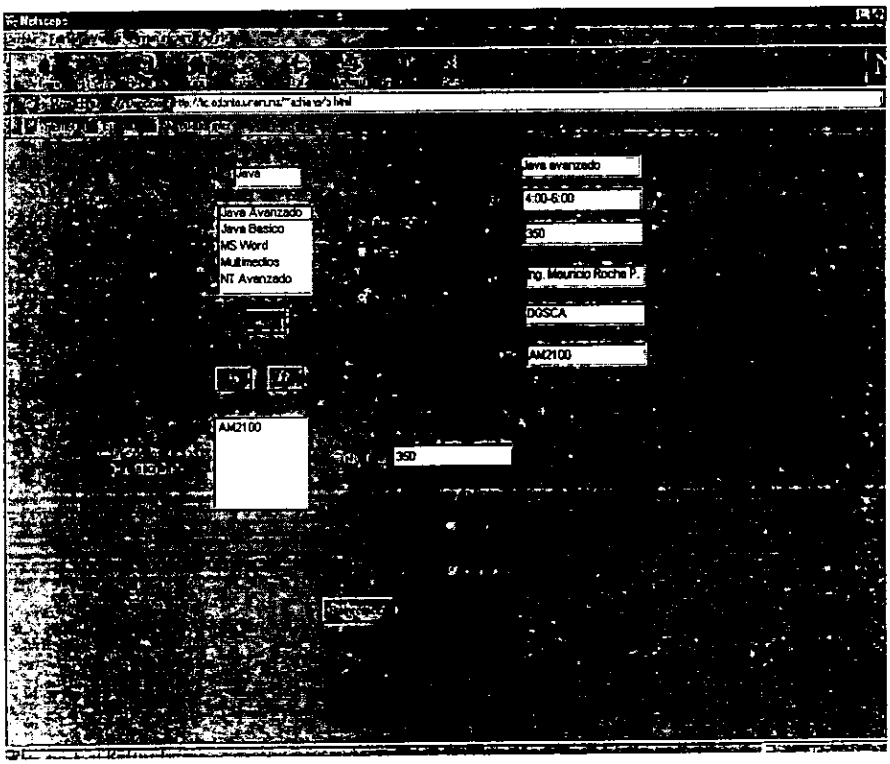
```
        lista.delItem(a);
        }

      while((linea=datos.readLine())!=null){
        k=0;l=0;
        for(int i=0;i<linea.length();i++){
         if('@'==linea.charAt(i)){
             Asignacion(linea.substring(l,i),j,k);
        k++;
        if(k==0)  l=i;
        else   l=i+1;
        }
        }//fin del if
      Asignacion(linea.substring(l,linea.length()),j,k);
      lista.addItem(tem.Valor_caja[j]);
      j=j+1;
      }
     }
   catch(IOException e){
    System.out.println("IO Error:" + e.getMessage());
    }
   }
   }//Fin de la clase Cursos
```

5.4 PÁGINA WEB

Ahora que se tienen la interfaz y las clases construidas, solo resta colocar el applet en la pagina web donde será consultado. El código necesario para realizar esto es el siguiente:

```
<html>
<p>
<center>
<applet code=curso.class width=500  height=500>
Tu visualizador no soporta JAVA.....
</applet>
</center>
</html>
```



CONCLUSIONES

Observando las características de java en términos generales podemos concluir lo siguiente:

- Un lenguaje no debe evaluarse solo desde un punto de vista técnico, considerando sintaxis y semántica, si no también por su contribución al desarrollo del software.

- Java ofrece grandes ventajas por características y méritos propios del lenguaje

- Es de reconocerse, el hecho de que java haya podido hacer que todos hagan una máquina virtual para su plataforma, dando así a java la característica de multiplataforma. Esto además de una característica intrínseca del lenguaje, es como haber logrado crear un estándar. Sin embargo, no debemos pasar por alto que como sucede en toda en la tecnología que se vende, sus fabricantes tratan de impulsarla, de meterla a toda costa en la mente del consumidor con la esperanza de colocarla en la lista de productos más vendidos.

- En términos generales el lenguaje tiene desventajas obvias como que el hecho de que la eficacia depende del aprendizaje, es decir, un buen programa en java depende del estudio que se tenga del propio lenguaje, de conocer la mayor cantidad de librerías.

- Tiene a su favor la gran cantidad de herramientas de apoyo programación y libros que explican e ilustran además de existencia de código.

- Para internet no hay duda que es una buena opción para apoyar a las páginas del web y facilitar al usuario su interacción con esta red.

El surgimiento de java ha sido un suceso en todo lo que concierne al desarrollo de sistemas distribuidos, pero aun existen muchas limitantes a pesar de las excelentes características del lenguaje; una de ellas es que todavía faltan varias plataformas sobre las cuales se tiene planeado que se ejecute Java. De esta manera las aplicaciones construidas en este lenguaje aún no son totalmente portables. Habrá que esperar el momento de vivir en un mundo de plataformas distintas, y que la red se vaya tornando en el "sistema operativo" a un nivel generalizado.

BIBLIOGRAFÍA

- OBJECT ORIENTED DESIGN WITH APLICATIONS

  Grady Booch

  Edit. Benjamin Cummnigs

- OBJECT ORIENTED ANALYSIS YOURDON PRESS

  Peter Cond

  Edit. Prentice-Hall

- CONECTATE AL MUNDO DE INTERNET

  Edie Kroll

  Edit. Mc Graw Hill

- MASTERING THE INTERNET

  Glee Harra Cady

  Edit. Sybex

- APRENDIENDO INTERNET EN 21 DÍAS

  Neil Randal

  Edit. Prentice-Hall

- SOFTWARE ORIENTADO A OBJETOS

  Ann L. Winblad, Samuel D. Edwars

  Edit. Addison Wesley / Diaz de santos

- ENTORNOS Y METODOLOGÍAS DE PROGRAMACIÓN

  F. Alonso Amo, F. J. Segovia Perez

  Edit. Parainfo

- OBJECT ORIENTED METHODS

  Ian Graham

  Edit. Addison Wesley

- APRENDIENDO JAVA EN 21 DÍAS

  Laura Lemay

  Edit. Prentice Hall

- HTML, JAVA, CGI, VRLM, SGML, WEB PUBLISHING UNLEASHED

  Wiliam Robert Stank

  Edit. Sams Net

- http:// cuiiwww.unige.ch/OSG/OOinfo/

- http://www.clark.net/pub/howie/OO/OO_home.html

- http://www.yahoo.com/Computers/Software/Object_Oriented_Programmin g/

- http://www.sun.com

- http://java.sun.com

- http://www.gamelan.com

- http://www.bib.udec.cl/manual.html#conceptos

- http://www.w3.org

- http://www.web.es

- http://www.drwebsa.com.ar/telnet.htm