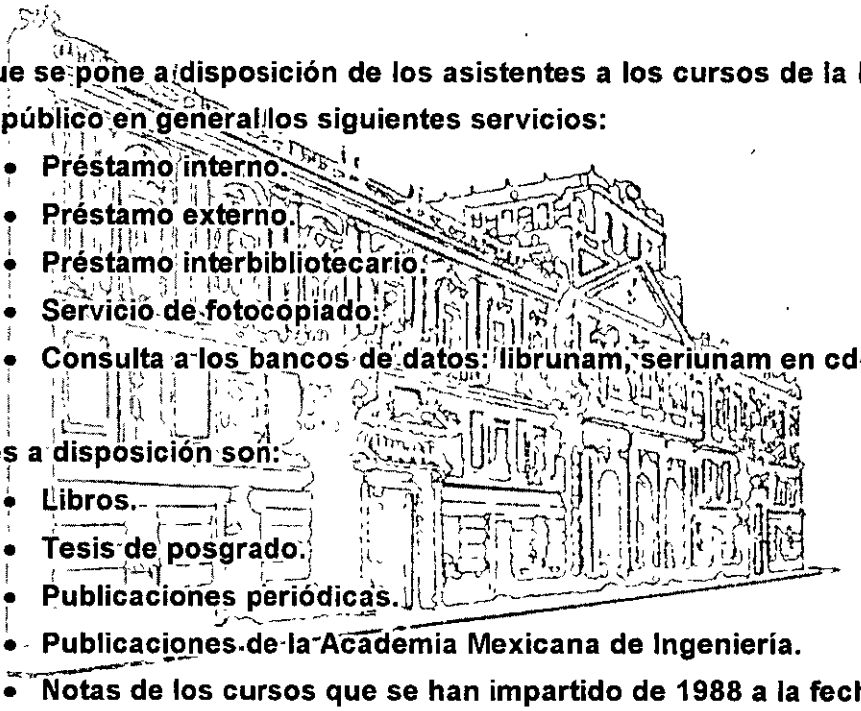
FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN
" ING. BRUNO MASCANZONI "

El Centro de Información y Documentación Ing. Bruno Mascanzoni tiene por objetivo satisfacer las necesidades de actualización y proporcionar una adecuada información que permita a los ingenieros, profesores y alumnos estar al tanto del estado actual del conocimiento sobre temas específicos, enfatizando las investigaciones de vanguardia de los campos de la ingeniería, tanto nacionales como extranjeras.



Es por ello que se pone a disposición de los asistentes a los cursos de la DECFI, así como del público en general los siguientes servicios:

- **Préstamo interno.**
- **Préstamo externo.**
- **Préstamo interbibliotecario.**
- **Servicio de fotocopiado.**
- **Consulta a los bancos de datos: librunam, seriunam en cd-rom.**

Los materiales a disposición son:

- **Libros.**
- **Tesis de posgrado.**
- **Publicaciones periódicas.**
- **Publicaciones de la Academia Mexicana de Ingeniería.**
- **Notas de los cursos que se han impartido de 1988 a la fecha.**

En las áreas de ingeniería industrial, civil, electrónica, ciencias de la tierra, computación y, mecánica y eléctrica.

El CID se encuentra ubicado en el mezzanine del Palacio de Minería, lado oriente.

El horario de servicio es de 10:00 a 14:30 y 16:00 a 17:30 de lunes a viernes.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

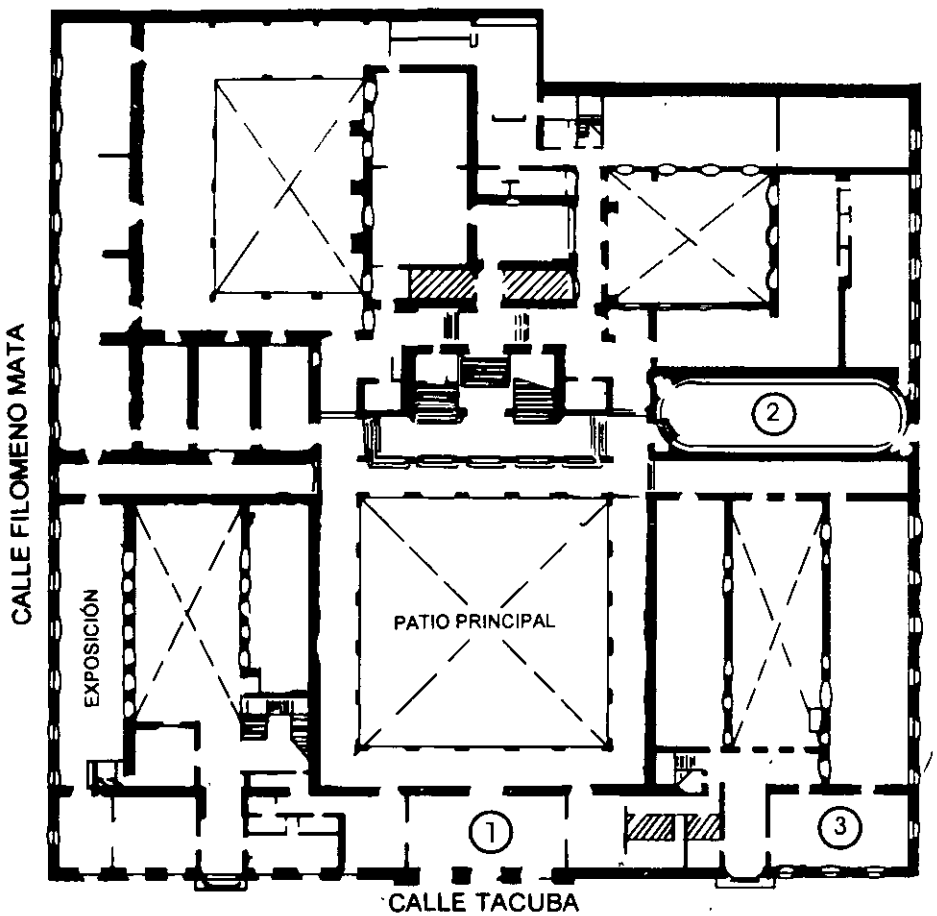
Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente

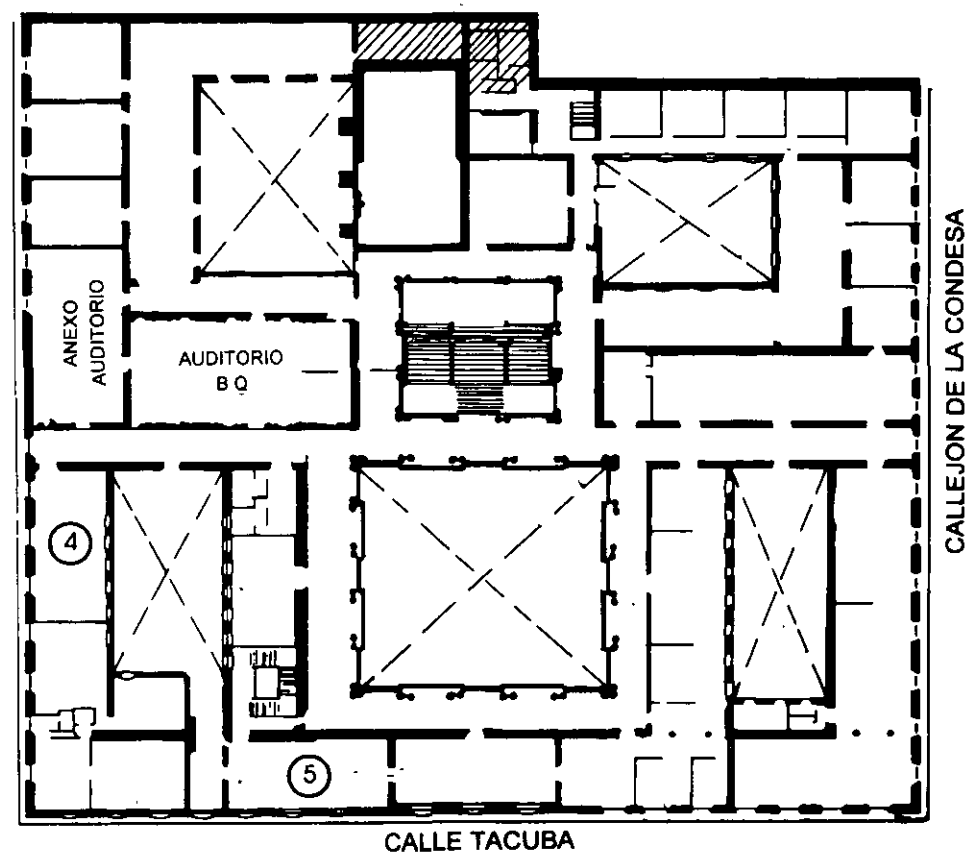
División de Educación Continua.

Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D F APDO Postal M-2285
Teléfonos 5512-8955 5512-5121 5521-7335 5521-1987 Fax 5510-0573 5521-4021 AL 25

PALACIO DE MINERIA

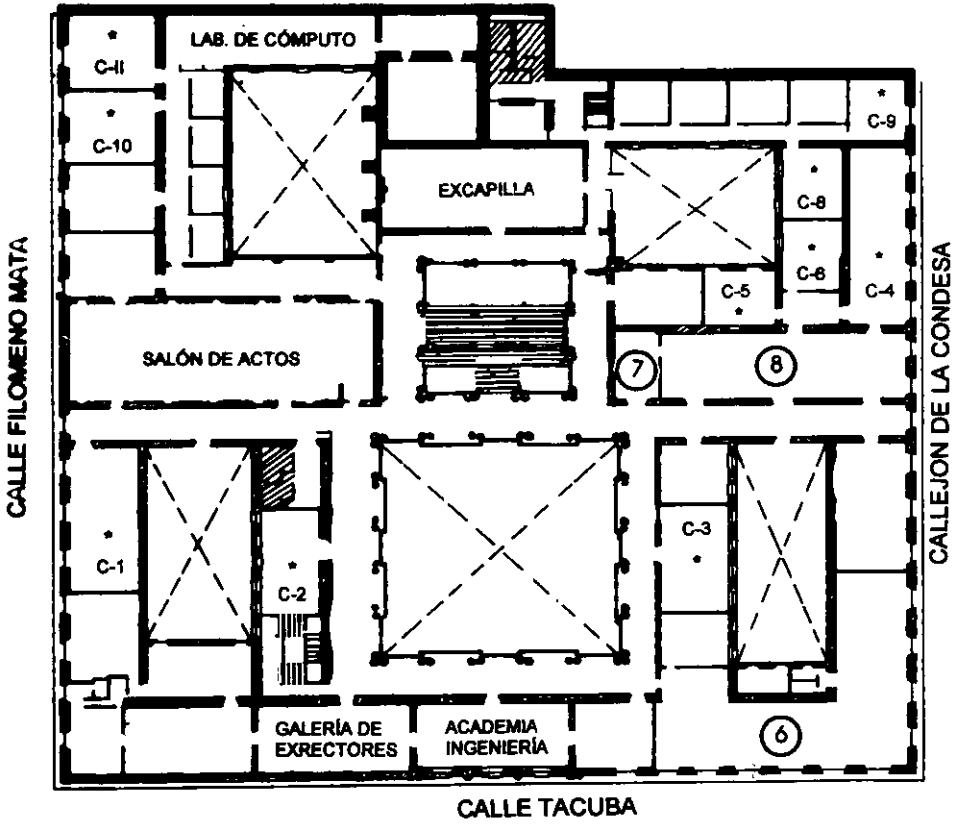


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA
DEC

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA

CURSO: ***CC013 Animación de Páginas WEB con Java y Java Script***
FECHA: ***16 al 25 de abril del 2001***

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACION 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
JONATHAN GAMBOA BELTRAN				
YAZMIN VEGA HERNANDEZ				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF
ORGANIZACION Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACIÓN DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1. ¿Le agradó su estancia en la División de Educación Continua?

SI ☐ NO ☐

Si indica que "NO" diga porqué: _____

2. Medio a través del cual se enteró del curso:

Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cuál)	

3. ¿Qué cambios sugeriría al curso para mejorarlo?

_____

_____

_____

_____

_____

_____

4 ¿Recomendaría el curso a otra(s) persona(s) ?

SI ☐ NO ☐

5.¿Qué cursos sugiere que imparta la División de Educación Continua?

_____

_____

_____

_____

_____

_____

6. Otras sugerencias

_____

_____

_____

_____

_____

FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

"Tres décadas de orgullosa excelencia" 1971 - 2001

ANIMACION DE PAGINAS WEB CON JAVA Y JAVA SCRIPT

ABRIL DEL 2001

Palacio de Minería, Calle de Tacuba No. 5, Primer piso, Delegación Cuauhtémoc, CP 06000, Col Centro, México D.F.,
APDO Postal M-2285 • Tels· 5521 4021 al 24, 5623 2910 y 5623 2971 • Fax 5510.0573

1 PROGRAMACION ORIENTADA A OBJETOS.

1.1 Historia de la Programación Orientada a Objetos.

En la actualidad la orientación a objetos se ha colocado como la corriente principal de la computación, tanto para los creadores de software como para los usuarios, y se emplea a lo largo de una amplia gama de componentes del software, incluyendo lenguajes, interfaces de usuario, bases de datos y sistemas operativos. Aunque la Orientación a Objetos no es la solución absoluta, ya ha demostrado que puede ayudar a gestionar la creciente complejidad y costos del desarrollo del software.

Al estar integrada en los componentes fundamentales del software, ha significado una revolución tan grande como la que represento la implementación de la programación estructurada en la década de los 70: un nuevo paradigma para mejorar la creación, mantenimiento y empleo del software. En este nuevo paradigma, los *objetos* y las *clases* son los pilares, mientras que los métodos, los mensajes y la herencia producen los mecanismos primarios. Históricamente, la creación de un programa implicaba la definición de procesos que actuaban sobre un conjunto independiente de datos. La Orientación a objetos cambia el centro de atención del proceso de programación desde el procedimiento hasta los *objetos* –módulos auto contenidos que incluyen tanto los datos como los procedimientos que actúan sobre dichos datos-.

La orientación a objetos no es un termino nuevo. De hecho tiene por lo menos 25 años de antigüedad. Sus raíces pueden encontrarse en Noruega a finales de los años 60 en conexión con el lenguaje llamado Simula67, desarrollado por Kristen Nygaard y Ole-Johan Dajl en el Centro de Cálculo Noruego. Simula67 introdujo por vez primera los conceptos de clases, y subclases, muy parecidos a los existentes en los lenguajes orientados a objetos de hoy en día.

Posteriormente, a mitad de la década de los 70, los científicos del Centro de Investigación de Palo Alto de Xerox crearon el lenguaje Smalltalk, el primer lenguaje orientado a objetos consistente y completo. En Smalltalk, cada elemento

del lenguaje fue realizado como un objeto. Incluso hoy, Smalltalk se considera el más puro de los lenguajes orientados a objetos. Los desarrollos Simula y Smalltalk impulsaron gran parte del trabajo que actualmente sucede en la orientación a objetos. Simula 67 demostró el poder de modelación de un lenguaje basado en clases, así como la idea de que los datos y operaciones deben de almacenarse juntos.

Simula67 y Smalltalk fueron relativamente inaccesibles a la corriente principal de la comunidad de las computadoras hasta los años 80. Por ejemplo, el trabajo inicial en Smalltalk no fue dado a conocer hasta el ejemplar de Agosto de 1981 de la revista *BYTE*. En los años 80 **C** se convirtió en un lenguaje de desarrollo muy popular, no sólo en las microcomputadoras sino en la mayoría de las arquitecturas y entornos de computación. Fue precisamente en esta década cuando Bjarne Stroustup de los Laboratorios Bell amplio el lenguaje **C** para crear **C++**, un lenguaje que soporta la programación orientada a objetos. Posteriores mejoras en herramientas y lanzamientos comerciales del lenguaje **C++**, justifican en buena parte el desarrollo general de la programación orientada a objetos, con **C++**, los programadores eran capaces de aprender al paradigma de la orientación a objetos en un léxico popular y conocido sin tener que invertir en nuevos y diferentes entornos y lenguajes de programación.

La orientación a objetos proporciona una mejor forma de gestionar la complejidad tecnológica.

1.2 Conceptos Básicos de la Programación Orientada a Objetos.

1.2.1 Objetos y Clases.

Un programa tradicional consta de procedimientos y de datos. Un programa orientado a objetos consta solamente de objetos que contienen tanto los procedimientos como los datos. Por decirlo de otra forma, los objetos son módulos que contienen los datos y las instrucciones que operan sobre esos datos. Así,

dentro de los objetos residen los datos de los lenguajes convencionales, como por ejemplo, números, matrices (arrays o arreglos), cadenas de caracteres y registros, así cualquier función, instrucción o subrutina que opere sobre ellos. Los objetos, por tanto, son entidades que tienen atributos (datos) y formas de comportamiento (procedimientos) particulares.

Muchos objetos diferentes pueden actuar de formas muy similares. Una *clase* es una descripción de un conjunto de objetos casi idénticos. Una clase consta de métodos y datos que resumen las características comunes de un conjunto de objetos. La posibilidad de abstraer métodos y descripciones de datos comunes de un conjunto de objetos y almacenarlos en una clase es esencial para la potencia de la orientación a objetos. Definir clases significa situar código reutilizable en un deposito común en lugar de volver a expresarlo una vez tras otra. En pocas palabras, las clases contienen los anteproyectos para crear objetos. Finalmente, la definición de una clase ayuda a clarificar la definición de un objeto: un objeto es un *modelo* o *ejemplar* de una clase.

Los objetos se crean cuando se recibe un mensaje solicitando su creación por la clase.

A diferencia de los elementos de datos pasivos en los sistemas tradicionales, los objetos tienen la posibilidad de actuar. La acción sucede cuando un objeto recibe un mensaje, que es, una solicitud que pide al objeto que se comporte de alguna forma. Cuando se ejecutan los programas orientados a objetos, los objetos reciben, interpretan y responden a mensajes procedentes de otros objetos. Los procedimientos o *métodos* que residen en el objeto, determinan como actúa el objeto cuando recibe un mensaje, manipulando las variables o *propiedades* del objeto.

1.2.2 Herencia.

La herencia es el mecanismo para compartir automáticamente métodos y datos entre clases, subclases y objetos. Este es un potente mecanismo que no se encuentra disponible en los lenguajes procedurales. La herencia permite a los

programadores crear nuevas clases programando solamente las diferencias con la clase padre.

Debido a la herencia, los programas orientados a objetos constan de taxonomías, árboles o jerarquías de clases que, por medio de la subclasificación, llegan a ser más especificas.

Herencia simple y múltiple son dos tipos de mecanismos de herencia utilizados normalmente en la programación orientada a objetos. Con la *herencia simple*, una subclase puede heredar datos y métodos de una clase simple así como añadir comportamiento por sí misma. La *herencia múltiple* se refiere a la posibilidad de una subclase de adquirir los datos y métodos de mas de una clase. La herencia múltiple es útil al construir comportamiento compuesto a partir de más de una rama de una jerarquía de clases.

1.2.3 Polimorfismo.

Los objetos actúan en respuesta a los mensajes que reciben. El mismo mensaje puede originar acciones completamente diferentes, al ser recibido por diferentes objetos. Este fenómeno se conoce como *polimorfismo*. Con el polimorfismo un usuario puede enviar un mensaje genérico y dejar los detalles exactos de su realización para el objeto receptor.

1.2.4 Abstracción.

La orientación a objetos fomenta que los programadores y usuarios piensen sobre las aplicaciones en términos abstractos. Comenzando con un conjunto de objetos, los programadores buscan un factor de comportamiento común y lo sitúan en superclases abstractas. Las bibliotecas de clases proporcionan un depósito de elementos comunes y reutilizables. Cada nivel de abstracción facilita el trabajo de programación porque hay disponible más cantidad de código reutilizable.

1.2.5 Encapsulación.

La *encapsulación* es el término formal que describe el conjunto de métodos y datos dentro de un objeto, de forma que el acceso a los datos se permite solo mediante los propios métodos del objeto. Es decir, al igual que las "cajas negras", la estructura interior de un objeto ésta oculta a los usuarios o incluso a los programadores. Los mensajes que recibe el objeto son los únicos conductos que conectan al objeto con el mundo exterior.

2. CARACTERISTICAS DE JAVA.

2.1 Historia.

El lenguaje Java es un lenguaje de propósito general orientado a objetos, diseñado por Sun Microsystems. Su desarrollo comenzó en Sun Microsystems, California en 1991. Inicialmente desarrollado para funcionar en una red de productos de consumo, se diseño para soportar múltiples arquitecturas, y para garantizar entregas seguras de componentes de software. Para cumplir estos requerimientos, el código compilado necesitaba ser transportado por la red, operar en cualquier tipo de cliente, y ofrecer seguridad al cliente, en su ejecución.

La popularización de la World Wide Web volvió estos atributos mucho más interesantes. La Internet demostró que era posible realizar contenido rico en medios, fácilmente accesible. Los primeros navegadores de red permitieron a millones de personas deambular por la red y hacer del WEB parte de la vida cotidiana. Por vez primera existía un medio donde lo que se veía o escuchaba era esencialmente lo mismo independientemente de la plataforma empleada, y sin importar el tipo de conexión.

Los entusiastas del WEB descubrieron muy pronto, que el contenido soportado por el formato de documento HTML era muy limitado. Las extensiones del mismo (como la inclusión de formas) solo hacían más notorias sus limitaciones, dejando claro que ningún navegador podría incluir todas las características que los usuarios anhelaban.

Hasta llegada del navegador de Sun, el HotJava que mostró al mundo las interesantes propiedades de la plataforma Java, haciendo posible la inclusión de programas, dentro de las páginas HTML. Estos programas son cargados en el navegador junto con la página a la que pertenecen. Antes de ser aceptados por el navegador, los programas son cuidadosamente examinados, para asegurarse de que fueran seguros. Así como las páginas HTML estos programas son independientes de la plataforma o de la red. Los programas se comportan de la

misma forma sin importar de que tipo de plataforma vienen, o en que tipo de maquina están siendo cargados.

La habilidad de ejecutar programas en pequeños dispositivos como los teléfonos celulares o los PDAs (Asistentes Digitales Personales) es una de las más valiosas propiedades de Java. Sun y otros fabricantes de hardware, descubrieron que la maquina virtual de java podía ser fácilmente colocada sobre los circuitos, en una amplia variedad de dispositivos. En la actualidad ya existen teléfonos celulares que corren Java. ¿Quizá algún día tengamos un refrigerador que corra Java?

2.2 La Maquina virtual de Java.

La maquina virtual de Java es la base de las plataformas Java y Java 2. Es el componente de esta tecnología, responsable por su independencia del Hardware y del Sistema Operativo, de lo diminuto que es el código compilado, y de su capacidad para proteger al usuario de programas dañinos

La maquina virtual, es la abstracción de una computadora. Como una computadora física, tiene un juego de instrucciones y manipula varias áreas de memoria en tiempo de ejecución.

El primer prototipo de la maquina virtual de Java, realizada en Sun Microsystems, era ejecutada en un dispositivo de mano, muy parecido a los actuales PDA. En la actualidad la maquina virtual de Java se encuentra disponible para todas las plataformas, ya sea PC, Mac o UNIX.

La MVJ no tiene relación directa con el lenguaje de programación, la tiene con un formato particular de archivos binarios, el formato *class*. Un archivo *class*, contiene las instrucciones de la maquina virtual (o bytecodes), así como otra información auxiliar.

Con fines de seguridad, la maquina virtual de Java, impone fuertes restricciones de formato y estructurales al código contenido en un archivo *class*.

2.3 Conceptos básicos de Java.

Para los fines de este curso existen algunas características esenciales de Java que será necesario conocer, para poder comprender el funcionamiento de nuestros Applets, y del lenguaje mismo.

2.3.1 Estructuras de control.

Siendo Java un lenguaje basado en C++ las sentencias de control de flujo son las mismas:

if / else
switch { case..... default }
for
while
do / while

así como el control general de flujo:

break
continue
return

2.3.2 Clases y objetos en Java.

En Java existen fundamentalmente dos tipos de clases a definir:

abstract

De una clase *abstract* (abstracta) no se pueden crear ejemplares (objetos), solo se emplea como base para la herencia.

final

Una clase *final*, es aquella que termina una rama de herencia. No se puede hacer herencia de una clase final.

La herencia entre clases, se implementa en Java mediante la palabra reservada:

extends

Por ejemplo, si se tiene la clase *figura_geometrica* se puede crear la subclase *circulo*, como una especialización de la clase *figura_geometrica*;

```
class circulo extends figura_geometrica {
    int radio;
   }
```

2.3.3 Modificadores de alcance.

Cuando se crea una nueva clase en Java, se especifica el nivel de acceso que se quiere para las variables y los métodos definidos en la clase:

public

```
public void nombre_del_metodo(){ }
```

Cualquier clase puede acceder a las variables y métodos declarados como públicos.

protected

protected void nombre_del_metodo(){ }

Solo las subclases pueden acceder a las variables y los métodos declarados como protected.

private

private nombre_de_la_variable;

Las variables y métodos declarados como privados sólo pueden ser accedidos desde dentro de la clase. No son accesibles desde las subclases.

amigables (sin declaración especifica)

nombre_del_metodo(){}

Por omisión si no se especifica el acceso, las variables y métodos de instancia son declarados como amigables (friendly), lo que significa que son accesibles para todas las clases dentro de su mismo paquete.

CAPITULO 3 APPLETS DE JAVA

Conforme empiece a programar con Java encontrará que el lenguaje tiene una estructura diseñada alrededor de la idea de los objetos. Tratando de conservar esta estrategia de diseño, muchos métodos que realizan funciones similares han sido agrupados en paquetes. Por ejemplo, el Abstract Window Toolkit es un paquete de métodos que son útiles para dibujar imágenes en la pantalla, trabajar con ventanas y construir interfaces de usuario. El Applet Package es un paquete que está diseñado específicamente para trabajar con applets.

El applet Package contiene varios métodos que están diseñados para usarse en la construcción de applets y en las circunstancias especiales que surgen con éstos. Por ejemplo, los applets necesitan ser capaces de cargar imágenes y audio clips desde un servidor, por lo que los métodos *getImage()* y *getAudioClip()* forman parte del Applet Package.

Algunos de los métodos del Applet Package.

MÉTODO	**FUNCIÓN**
public String getAppletInfo()	Devuelve información acerca del applet, como el nombre del autor.
public URL getDocumentBase()	Devuelve el URL del documento HTML.
public String getParameter (String name)	Devuelve los parámetros de un applet.
public String [][] getParameterInfo()	Devuelve un resumen de lo que controlan los parámetros.
public AudioClip getAudioClip(URL)	Se emplea para cargar un audio clip.
public Image getImage(URL)	Se emplea para cargar un archivo de imagen.
public void play (URL)	Se emplea para reproducir un audio clip cargado con anterioridad.

public boolean isActive()	Le permite saber si un applet está activo.
public void resize (int, int)	Se emplea para modificar el tamaño de un applet.
public void showStatus(String msg)	Despliega una cadena de estado en el visualizador del applet.
public void init()	Inicializa el applet.
public void start()	Arranca el applet una vez que ha terminado de inicializarse.
public void stop()	Detiene el applet cuando usted abandona la página de éste.
public void destroy()	Destruye el applet cuando usted abandona el visualizador.

3.1 COMO FUNCIONAN LOS APPLETS.

Un applet es un programa compilado basado en el código fuente que se ejecuta a través del navegador.

El código de su applet tiene que llamarse algo.java para que pueda ser compilado correctamente y el archivo resultante recibe el nombre de algo.class. Estas extensiones le ayudan a distinguir entre el código y los applets compilados, pero también contribuyen a la estructura de clases. Cada applet puede funcionar como un objeto y por lo tanto es necesario estructurarlo de manera que pueda usarse como un objeto. Es por esto que un applet compilado tiene un nombre en la forma de algo.class. la extensión class hace saber al compilador de Java que la información contenida en este archivo es una definición de clase.

3.1.1 El Ciclo de vida de los applets.

Cuando se carga una página WEB que contiene un applet, este pasa por varias etapas durante el tiempo que lo vemos en pantalla. El applet realiza tareas muy

diferentes en cada una de estas etapas, aunque la mayoría de dichas tareas no son visibles para el usuario final. Las etapas son inicialización, ejecución y terminación.

init()

El primer método llamado por un applet después de que ha sido cargado por el visualizador es *init()*. Debido a que el applet no está siendo ejecutado en el momento en el que se llama al método *init()*, este método es un lugar excelente para encargarse de cualquier tare básica que deba realizarse para que el applet que reproduce un audio clip al hacer clic en un botón. En un applet como éste, sería necesario cargar el audio clip y configurar el botón antes de que el applet comience a ejecutarse. Para realizar estas tareas sería necesario cargar el audio clip y configurar el botón antes de que el applet comience a ejecutarse.

start ()

Después de que el applet ha sido cargado en el visualizador y está listo para empezar, se llama automáticamente al método *start()*. Por lo general, este método contiene la esencia de los apletts. Después de todo, este método es lo que usted quiere que su applet haga. En la mayoría de los applets es necesario definir métodos *init()* y *start()*.

stop()

El método *stop()* es la contraparte del método *start()*. Se llama automáticamente cuando un applet debe detener su ejecución, cuando se abandona la página WEB del applet, por ejemplo si el método *start()* para iniciar algunas funciones que sea necesario detener antes de que el usuario siga adelante, debe emplear el método *stop()* para detenerlas.

destroy()

El método *destroy()* es, en esencia. La muerte de un applet. Cuando abandona su visualizador, este método es invocado automáticamente para realizar cualquier limpieza que sea necesaria. Tal y como lo implica su nombre, el método *destroy()* elimina todo rastro de su applet. Purga cualquier espacio de memoria que haya sido utilizado por su applet y detiene los hilos o métodos activos. Hablando en términos generales, no tiene que hacer nada para usar el método destroy(); un método *destroy()* base es predefinido e invocado de manera automática.

Ejemplo.

El siguiente ejemplo es un applet que reproduce un audio clip al hacer clic en un botón.

```
import java.applet.*;
import java.awt.*;                    //importación de librerías
import java.lang.*;
import java.net.URL;

public class Audio extends java.applet.Applet {        //declaración de la clase

        AudioClip clip;

// en un applet como éste, es  necesario cargar el audio clip y configurar el
botón antes de que le applet comience a ejecutarse.

public void init() {                  //inicia método init()
        clip= getAudioClip(getDocumentBase(), Soundfile.au);
```

```
        setLayout(new FlowLayout());
        play=new Button("Play clip");
        add(play);
}       //termina método init()
```

//este método que usted quiere que haga.

```
public void start() {              //inicia método start()
        clip.play();
}               //termina método start()
```

//en este ejemplo todo lo que hace el método stop es detener la reproducción del audio clip .

```
public void stop() {               //inicia método stop()
        clip.stop();
}                       //termina método stop()


}
```

3.2 HTML y Java

En realidad sólo existen dos etiquetas básicas **<APPLET>** y **<PARAM>**. Así al agregar las etiquetas HTML básicas y adaptar algunas de las ya existentes, usted puede especificar toda la información que un visualizador necesita para ejecutar su applet.

3.2.1 LA ETIQUETA <APPLET>

Agregar un applet a su página Web es simplemente cosa de emplear la etiqueta <APPLET>. Esta etiqueta le hace saber al navegador que usted proporcionará información especifica acerca de un applet de Java que desea que aparezca en su página. La etiqueta <APPLET> también acepta una variedad de atributos, como *code, width y heigth,* los cuales le permiten personalizar la apariencia que el applet tendrá en sus páginas.

Estructura básica:

```
<html>
<applet code=nombredearchivo.class  height=n width=m>
</applet>
</html>
```

codebase=URL

codebase es la ubicación URL base de su applet Java. Cuando se emplea el atributo *code=nombredearchivo.class*, el *nombredearchivo* se refiere ya sea al documento actual o a un *codebase* especifico. Especificar un *codebase* le permite tener su código en un directorio diferente al que contiene su applet, lo cual puede resultar útil para reutilizar applets en diferentes páginas en su servidor.

```
<applet  codebase="classes" code="tickertape.class" width=500 height=59>
```

El atributo *codebase* le hace saber al visualizador que el applet está almacenado en el directorio "classes". El atributo code proporciona el nombre del archivo del applet.

Width y Height

Los atributos *width* y *height* están diseñados para permitirle especificar las dimensiones iniciales de su applet en píxeles.

Name=nombre

El atributo *name* le permite especificar un nombre simbólico (como un sobrenombre) para un applet. Si está diseñando sus applets de manera que se ensamblen unos sobre otros por razones de funcionalidad o requiere de otro applet para procesar información, los applets necesitan alguna manera de identificarse entre sí para poder intercambiar información. Este atributo le permite nombrar su applet para que otros applets puedan comunicarse con él por su nombre.

<applet codebase="classes" code="tickertape.class" name="ticker" width=500 height=59>

align=[center, left, right]

El atributo *align* le permite alinear con respecto al navegador el applet.

<applet codebase="classes" code="tickertape.class" width=500 height=59 align="right">

vspace=n y hspace=m

Cuando emplee el atributo align para colocar un applet, tal vez quiera asegurarse de que ningún otro elemento se superponga o invada el área del applet. Los atributos *vspace* y *hspace* le permiten especificar en píxeles un espacio en búfer

horizontal y/o vertical alrededor de un applet para que los otros elementos guarden una distancia decente en relación con el applet.

<applet codebase="classes" code="tickertape.class" width=500 height=59 vspace=25 hspace=25 >

alt="texto"

Al diseñar páginas WEB con Java, tome en cuenta la posibilidad de que a que observe su página no cuente con un visualizador con capacidad Java. El atributo alt le permite especificar una imagen, texto o URL que podrá ser observado en lugar de su applet de Java.

<applet codebase="classes" code="tickertape.class" width=500 height=59 alt="lo siento, necesita un visualizador Java para observar este applet">

3.2.2 LA ETIQUETA <PARAM>

Muchos applets existen para ofrecer características personalizadas en una página WEB y para estos applets usted podría incluir todos los parámetros en el código.
Al controlar estos atributos con parámetros en lugar de codificarlos, el applet se vuelve se vuelve más robusto y flexible, permitiendo que más gente agregue el applet a sus propias páginas base.
La etiqueta <PARAM> se emplea para pasar parámetros al applet. De hecho, la etiqueta <PARAM> se debe usar con la etiqueta <APPLET> para que tenga algún sentido, por lo que en cierta forma puede decirse que es un atributo de la etiqueta <APPLET>. La etiqueta <PARAM> es una etiqueta con dos atributos: *name* y *value*.

El atributo *name* le permite especificar el nombre del parámetro y le atributo *value* le permite especificar el valor correcto para dicho parámetro. La etiqueta <PARAM> no requiere una etiqueta final. Un applet requiere de una etiqueta <PARAM> para cada parámetro que se desee determinar.

4.¿QUÉ ES JAVASCRIPT?

JavaScript es la más cómoda y sencilla posibilidad de diseñar óptimamente documentos Web. Dada la facilidad con que se trabaja con JavaScript por su flexibilidad y escasez de reglas, se le puede considerar como un lenguaje de archivos de comandos, dándose a entender como una versión "light" de un lenguaje de programación.

Javascript amplía las capacidades de una pagina WEB estándar, mucho más allá de sus posibilidades normales. Con JavaScript se puede dar respuesta a eventos iniciados por el usuario (el observador de nuestras páginas, por ejemplo), eventos tales como la entrada de una forma o algún enlace. Esto sucede sin ningún tipo de transmisión. De tal forma que cuando un usuario escribe algo en una forma, no es necesario que sea transmitido hacia el servidor, verificado y devuelto. Las entradas son verificadas por la aplicación cliente y pueden ser transmitidas después de esto.

JavaScript está controlado por eventos, siempre que sucede algo en una página WEB, se produce un evento. Evento puede ser todo: la pulsación sobre un botón, el movimiento del puntero del ratón, cuando se carga una página, etc. JavaScript está controlado por eventos, de forma que reaccionará ante la aparición de cualquier evento. El tipo de reacción dependerá de la forma en que este programado.

JavaScript es independiente de la plataforma. Como se sabe un programa que funciona bajo Windows no se puede ejecutar en un Macintosh (a no ser que el Macintosh simule el entorno Windows). Sin embargo, JavaScript no tiene dependencia funcional bajo ninguna plataforma y solo está vinculado al navegador que lo interpreta el mismo JavaScript.

4.1 JAVA Y JAVASCRIPT.

Aunque JavaScript y Java, trabajan en el mismo entorno la –WWW- y ambos son lenguajes apropiados para crear aplicaciones de Internet, no son lo mismo. La

primera diferencia empieza porque los dos productos proceden de fabricantes distintos: Java procede de Sun y JavaScript de Netscape. Aunque existen acuerdos entre Netscape y Sun en lo que respecta a la seguridad de ambos productos; no obstante, éstos se desarrollaron independientemente. Otra diferencia entre Java es un lenguaje orientado a objetos, esto es, que el programador puede crear sus propios objetos, y JavaScript es un lenguaje basado en objetos esto es que los objetos están integrados en el lenguaje.

Algo que debe quedar claro es que JavaScript no es un subconjunto de Java ni viceversa, ambos son entidades independientes que, en ciertos aspectos, presentan algunas similitudes, pero que básicamente persiguen objetivos distintos.

Algunas de las principales diferencias entre Java y JavaScript

JavaScript	Java
Se interpreta por el cliente.,	Se compila por el programador y se ejecuta por el cliente.
El código está incrustado en HTML	El código (applet) se carga como módulo.
Utilización sencilla de datos / tipos de datos.	Reglas muy estrictas para los tipos de datos (se tienen que declarar)
Código sólo utilizable en documentos HTML.	Permite la creación de programas independientes.
Colabora con elementos HTML.	Supera las capacidades de HTML
Posibilita el acceso directo a los objetos del navegador.	Carece de acceso a los objetos del navegador y a su funcionalidad.

Basado en objetos: se pueden utilizar los objetos integrados, pero carece de creación de clases y de herencia.	Orientado por objetos: los applets constan de clases con herencia.
Vínculo dinámico: las referencias de objetos se verifican en tiempo de ejecución.	Vínculo estático: las referencias de objetos tienen que estar establecidas ya al efectuarse la compilación.

4.2 OBJETOS EN JAVASCRIPT

JavaScript pone a nuestra disposición. Estos objetos se pueden dividir en tres grupos:

- Objetos integrados.
- Objetos HTML.
- Objetos del navegador.

4.2.1 OBJETOS INTEGRADOS.

A los objetos integrados pertenecen los objetos string, date y math. Estos objetos se denominan integrados porque no tienen nada que ver con las páginas WEB, con las definiciones HTML, las direcciones URL, el entorno del navegador ni con cualquier otro elemento visible.

4.2.2 OBJETOS HTML.

Todo vínculo y cada ancla son un objeto JavaScript. Cada formulario y elemento del mismo es un objeto HTML. La estructura jerárquica de una página Web se refleja en la ordenación de los objetos HTML anidados.

4.2.3 LOS OBJETOS DEL NAVEGADOR

Los objetos del navegador se sitúan en la cima de la jerarquía de los objetos JavaScript. Estos objetos representan los elementos del entorno actual del navegador. A ellos pertenecen los objetos window, history y location.

Como cualquier otro lenguaje de programación, JavaScript también establece vínculos. Estos vínculos o métodos manipulan la información con la ayuda de objetos. JavaScript esta limitado a trabajar con los objetos del navegador. Esto le permitirá crear nuevos documentos y modificar los formularios existentes.

Aunque esto es una limitación ya que no se pueden agregar capacidades multimedia, usando exclusivamente JavaScript, para agregarlas hay que utilizar applets de Java.

OBJETO DE JAVASCRIPT	CONTROLADOR DE EVENTOS
Cuadro de lista	onBlur, onChange, onFocus
Documento	onLoad, onUnload.
Ventana	onLoad, onUnload.
Formulario	OnSubmit
Vínculo	onClick, onMouseOver
Casilla de verificación	OnClick
Cuadro de opción	OnClick
Botón-Reset	OonClick
Elemento de botón	OonClick
Botón-Submit	OnClick
Elemento área de texto	onBlur, onChange, onFocus, onSelect
Cuadro de texto	onBlur, onChange, onFocus, onSelect

Descripción de eventos

EVENTO	DESCRIPCIÓN
Blur	Se genera cuando se quita el enfoque de un elemento de formulario, es decir cuando el usuario pulsa el botón fuera del campo del elemento.
Clic	Se genera cuando el usuario pulsa un vínculo o un elemento de formulario
Change	Generado cuando el usuario modifica el valor de un elemento de formulario.
Focus	Se genera cuando el enfoque se encuentra en un elemento de formulario.
Load	Generado al cargár una página en el navegador.
Mouseover	Se genera cuando el usuario coloca el puntero del ratón en un hipervínculo.
Select	Generado al seleccionar dentro de un elemento de formulario.
Submit	Generado al activar un formulario pulsando el botón Submit
Unload	Se genera cuando el usuario abandona una página.

BIBLIOGRAFÍA.

EL GRAN LIBRO DE JAVASCRIPT.
Kolbeck Rainer.
Marcombo, Barcelona 1997.

CREE SUS APPLETS PARA WEB CON JAVA.
Gulbransen David y Rawlings.
Prentice Hall Hispanoamericana, México 1996.

COMO PROGRAMAR EN JAVA
Deitel y Deitel.
Prentice Hall Hispanoamericana, México 1998

Software orientado a objetos.
Ann L. Winbland, Samuel D. Ewards, David R. King.
Editorial Addison-Wesley 1993
Delaware, USA.

Hacking JAVA.
Mark Wutka.
Editorial QUE 1997
USA.

The Java Virtual Machine Specification.
Tim Lindholm, Frank Yellin.
Sun Microsystems 1999
USA.