"2019, Año del Caudillo del Sur, Emiliano Zapata"

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

MANUAL DE PRACTICAS DE PROGRAMACIÓN EN AMBIENTE CLIENTE SERVIDOR.

ELABORADO POR:

JOSÉ MADRID MARTÍNEZ

DICTAMEN: AS-2-105/2018

Reporte Final del periodo comprendido del 13 de agosto de 2018 al 12 de agosto de 2019.

Hermosillo, Sonora agosto 2019

2.- AGRADECIMIENTOS

Agradezco primeramente a mi familia, a mi esposa Cecilia y a mi hija Diana Denisse por su apoyo, comprensión y paciencia para la realización de mi año sabático, al departamento de desarrollo académico por su asesoría en el llenado de documentación requerida para la realización del trámite, a la dirección del Instituto Tecnológico de Hermosillo y a su director MCI Carmen Adolfo Rivera Castillo por su apoyo para la obtención de la prestación así como también al Tecnológico Nacional de México por la autorización del año sabático y a todas aquellas personas que de alguna forma se han involucrado con este proyecto.

Agradezco también la oportunidad que me ha dado la vida de participar en esta maravillosa y noble tarea que es la de enseñar, ser maestro de una Institución de Educación Superior, es una de las profesiones más nobles que existen, es una gran responsabilidad porque se tiene en las manos la facultad de formar positivamente a las nuevas generaciones de profesionista que impactaran en nuestra sociedad, pero lo más importante para mí es el privilegio de aprender de ellos y saber que no dejare de hacerlo porque mientras más enseño a mis alumnos, es más lo que yo aprendo de ellos.

No puedo dejar de agradecer la inmensa fortuna que tengo al enseñar en este mágico y cambiante mundo de las ciencias computacionales, en comprender como han evolucionado y evolucionan día con día las nuevas tecnologías de la información es algo grandioso, como docente se tiene la obligación de auto capacitarse para estar al corriente en las fronteras del conocimiento de nuestra disciplina de estudio, pero como docente del área de sistemas computacionales es aún más apremiante por los cambios tan rápidos y drásticos que se presen, ya no día con día, sino hora tras hora.

El trabajar en el desarrollo del proyecto de año sabático, me ha dado la oportunidad de capacitarme más en la disciplina que tanto me apasiona, la COMPUTACION. ***GRACIAS.***

3.- INDICE *Página.*

4- PRESENTACIÓN

4.1- OBJETIVO GENERAL DEL MANUAL.

Fomentar entre los estudiantes actividades prácticas que promuevan el desarrollo de habilidades en el manejo del análisis y diseño de soluciones de aplicaciones distribuidas utilizando los principales middlewares para el desarrollo de sistemas cliente/servidor.

Actuar como facilitador, orientador, inductor del aprendizaje, facilitando dicho proceso a través del uso del manual de prácticas de, ya que esta materia cuenta en sus créditos con horas prácticas.

4.2.- OBJETIVOS ESPECIFICOS DEL PROYECTO.

1. Facilitar al estudiante los materiales de consulta de la asignatura y relacionar los conocimientos teóricos con los prácticos con la finalidad específica de ser utilizado como un medio didáctico para la materia y apoyar con esto el proceso enseñanza - aprendizaje para el logro de las competencias específicas de la materia.
2. Promover la responsabilidad del estudiante en la conducción de su aprendizaje ya que nuestro modelo educativo está centrado en el aprendizaje.
3. Promover el aprendizaje colaborativo entre los estudiantes con la obtención de conclusiones de las prácticas desarrolladas por parte de los equipos integrados para la realización de la misma.
4. Complementar el aprendizaje de los contenidos abordados en el aula.

4.3.- JUSTIFICACIÓN

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Uno de los ejemplos más “antiguos” en este contexto es el correo electrónico, que demuestra a cada segundo de su funcionamiento los principios del modelo cliente servidor. En este caso, el cliente envía y recibe mensajes que “viajan” a través de redes de comunicación, y éstos se alojan en “buzones” cuyo nombre técnico es servidores de correo.

Algunos otros ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son un Servidor de impresión y la World Wide Web.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los *sistemas multicapa* en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes **computadoras** aumentando así el grado de distribución del sistema.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

4.4.- UTILIDAD DEL MANUAL

Un manual de prácticas adquiere una profunda relevancia en el ámbito del proceso de enseñanza aprendizaje, de tal manera se ubica como elemento esencial en el proceso educativo. En este contexto se debe entender primordialmente que el manual de prácticas de Programación Ambiente Cliente Servidor es una herramienta que permite por un lado la adecuada planeación docente de la asignatura y por otro lado que el estudiante tenga un papel activo, reflexivo y participativo en el proceso de enseñanza-aprendizaje.

Podemos decir que el manual de prácticas es un elemento fundamental en al llevar la teoría a la práctica, logrando que los estudiantes experimenten directamente en los equipos computacionales los diferentes modelos que se presentan en el desarrollo de los sistemas distribuidos.

Este trabajo se ubica en el marco de las clases prácticas (3-2-5 de acuerdo con el **Sistema de Asignación y Transferencia de Créditos Académicos**), el tipo de clase que tiene como objetivos instructivos fundamentales que los estudiantes ejecuten, amplíen, profundicen, integren, y generalicen determinados métodos de trabajo de las asignaturas, que les permita desarrollar habilidades para utilizar y aplicar, de modo independiente, los conocimientos".

Así entonces este manual de prácticas es un elemento básico para el trabajo experimental, el cual se lleva a cabo durante las clases prácticas, mismas que se desarrollan en el laboratorio del Centro de Cómputo del Tecnológico que es el lugar de trabajo, en la enseñanza y en la investigación, en donde se realizan desarrollos y practicas sobre la arquitectura Cliente Servidor.

Para adquirir un conocimiento adecuado sobre el tema se ha desarrollado este manual de prácticas para permitirle al estudiante de Ingeniería en Sistemas adquirir las destrezas necesarias en el manejo de la programación para su desempeño profesional.

4.5.- PROGRAMA DE LA MATERIA.

1.- DATOS DE LA ASIGNATURA

Nombre de la asignatura**:**	**Programación en Ambiente Cliente/Servidor**
Carrera:	**Ingeniería En Sistemas Computacionales**
Clave de la asignatura:	**IFF-1019**
SATCA[1:]	**3-2-5.**

[1] Sistema de Asignación y Transferencia de Créditos Académicos

2.- PRESENTACIÓN

Caracterización de la asignatura.
Esta asignatura aporta al perfil del Ingeniero en Informática la capacidad para analizar, diseñar y desarrollar aplicaciones distribuidas que atiendan y resuelvan las necesidades de información de las organizaciones.

Intención didáctica.
En la primera unidad se analiza el modelo de programación cliente/servidor y se describen las características principales de los modelos de computación distribuida.

De los conocimientos básicos para la creación de una interfaz socket se encarga la unidad dos; aquí los estudiantes identifican las características de los sockets y las utilizan en el desarrollo de aplicaciones cliente/servidor.

En la tercera unidad el estudiante se introduce al conocimiento de modelos computacionales distribuidos analizando el uso de la invocación remota de métodos RMI usando el lenguaje de programación Java, para desarrollar una aplicación cliente/servidor.

El modelo de objetos componentes COM/DCOM es otro modelo computacional distribuido con propósitos semejantes al RMI que está orientado en ambientes propios de Windows.

3.- COMPETENCIAS A DESARROLLAR

Competencias específicas:	**Competencias genéricas:**
• Conocer los distintos aspectos de programación cliente servidor. • Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor. • Utilizar sockets en el desarrollo aplicaciones cliente/servidor • Seleccionar el modelo de cómputo distribuido pertinente para una aplicación específica. • Crear e implementar un servicio web.	<u>Competencias instrumentales</u> • Capacidad de análisis y síntesis. • Capacidad de resolver. • Conocimientos básicos de la carrera. • Comunicación oral y escrita. • Habilidades avanzadas de manejo de la computadora. • Habilidad para buscar y analizar información proveniente de fuentes diversas. • Solución de problemas. • Toma de decisiones. Competencias interpersonales • Capacidad crítica y autocrítica. • Trabajo en equipo. • Habilidades interpersonales. <u>Competencias sistémicas</u> • Capacidad de aplicar los conocimientos en la práctica. • Habilidades de investigación. • Capacidad de aprender. • Capacidad de generar nuevas ideas (creatividad). • Capacidad para diseñar y gestionar proyectos. • Habilidad para trabajar en forma autónoma. • Búsqueda del logro.

4.- HISTORIA DEL PROGRAMA.

Lugar y fecha de elaboración o revisión	Participantes	Evento
	Representantes de los Institutos	

Instituto Tecnológico de Saltillo del 5 al 9 de octubre de 2009.	Tecnológicos de: Apizaco, Cerro Azul, Chetumal, Ciudad Juárez, Ciudad Madero, Superior de Coatzacoalcos, Colima, Comitancillo, Conkal, Durango, El Llano Aguascalientes, El Salto, Superior de Fresnillo, Huejutla, Superior de Lerdo, Linares, Los Mochis, Mexicali, Morelia, Oaxaca, Superior del Occidente del Estado de Hidalgo, Ocotlán, Orizaba, Piedras Negras, Pinotepa, Saltillo, San Luis Potosí, Tapachula, Tijuana, Torreón, Tuxtepec, Superior de Valladolid, Valle del Guadiana, Superior de Zacapoaxtla y Zacatecas.	Reunión Nacional de Diseño e Innovación Curricular para el Desarrollo y Formación de Competencias Profesionales de la Carrera de Ingeniería Informática.
Desarrollo de Programas en Competencias Profesionales por los Institutos Tecnológicos del 12 de octubre de 2009 al 19 de febrero de 2010.	Academias de Ingeniería Informática de los Institutos Tecnológicos de: El Llano, Colima y Los Mochis.	Elaboración del programa de estudio propuesto en la Reunión Nacional de Diseño Curricular de la Carrera de Ingeniería Informática.
Instituto Tecnológico de Zacatecas del 12 al 16 de abril de 2010.	Representantes de los Institutos Tecnológicos de: Apizaco, Cerro Azul, Chetumal, Ciudad Juárez, Ciudad Madero, Superior de Coatzacoalcos, Colima, Comitancillo, Conkal, Durango, El Llano Aguascalientes, El Salto, Superior de Fresnillo, Huejutla, Superior de Lerdo, Los Mochis, Mexicali, Morelia, Oaxaca, Superior del Occidente del Estado de	Reunión Nacional de consolidación de los Programas en Competencias Profesionales de la Carrera de Ingeniería

	Hidalgo, Ocotlán, Orizaba, Piedras Negras, Pinotepa, Saltillo, San Luis Potosí, Tapachula, Tijuana, Torreón, Tuxtepec, Superior de Valladolid, Valle del Guadiana, Superior de Zacapoaxtla y Zacatecas.	Informática.

5.- OBJETIVO GENERAL DEL CURSO

Analiza y diseña soluciones de aplicaciones distribuidas utilizando los principales middlewares para el desarrollo de sistemas cliente/servidor.

6.- COMPETENCIAS PREVIAS

- Lenguaje de programación Java, Visual C o Visual Basic
- Conocimientos básicos en sistemas operativos como Windows y Linux
- Conocimiento del protocolo de red TCP/IP.

7.- TEMARIO

Unidad	Temas Subtemas	Temas Subtemas
1	Contexto de la programación cliente/servidor	1.1. Arquitectura del modelo cliente/servidor. 1.2. Modelos de dos y tres capas. 1.3. Usos y aplicaciones 1.4. Comunicación entre programas 1.5. Modelos de computación distribuida 1.5.1. RMI 1.5.2. COM/DCOM. 1.5.3. Servicios Web. 1.5.4. Otros. .
2	Programación ClienteServidor de Bajo Nivel: sockets y canales	2.1. Concepto de socket. 2.2. Dominios y Tipos de sockets. 2.3. Creación/ implementación y supresión de sockets. 2.4. Desarrollo del lado del servidor con sockets. 2.5. Desarrollo del lado del cliente con sockets.
3	RMI (REMOTE METHOD INVOCATION)	3.1. Características y Estructura de RMI 3.2. El API Java RMI. 3.3. Jerarquía de objetos RMI. 3.4. El Sistema de Nombrado Registry. 3.5. Desarrollo de Aplicaciones Distribuidas. 3.6. Paso de parámetros a través de la red. 3.7. Callbacks (Resguardos).

4	COM/DCOM (Component Object Model/ Distributed COM)	4.1. Creación de Servidores COM. 4.2. Creación de un cliente COM. 4.3. Automatización. 4.4. ATL (Active Template Library). 4.5. DCOM (Distributed COM).
5	Servicios web XML	5.1. Características del lenguaje 5.2. Visión general de servicios web XML 5.3. Tecnologías subyacentes. 5.3.1. SOAP 5.3.2. WSDL 5.3.3. UDDI

8.- SUGERENCIAS DIDÁCTICAS

El docente debe:

- Fomentar el uso de las tecnologías de información y comunicación.
- Dar cabida a la flexibilidad en el seguimiento del proceso formativo y propiciar la interacción entre los estudiantes.
- Reforzar la integración y continuidad de los equipos de trabajo; propiciar la realización de investigaciones de campo.
- Tomar en cuenta el conocimiento de los estudiantes como punto de partida y como posible obstáculo para la construcción de nuevos conocimientos.
- Hacer que el estudiante se ubique en la realidad al indagar sobre las experiencias tecnológicas del ambiente externo en que se desenvuelve. Ejemplos: casos relacionados con el uso de aplicaciones de red desarrolladas en el paradigma cliente/servidor.
- Relacionar los contenidos de esta asignatura con los obtenidos en las demás del plan de estudios, reforzando la importancia de tener una visión y práctica interdisciplinaria para alcanzar las metas académicas, profesionales y empresariales.
- Motivar el desarrollo de capacidades intelectuales relacionadas con la escritura, la expresión oral y la lectura de documentos técnicos. Ejemplo: Redactar cada uno de los productos que se marcan como actividades de aprendizaje.
- Orientar al estudiante en la preservación del medio ambiente, al ver que cumpla con la normatividad relativa con la minimización del impacto ambiental negativo, al momento de realizar una innovación o cambio tecnológico en alguna empresa.
- Cuando los temas lo requieran, utilizar medios audiovisuales para una mejor comprensión del estudiante.
- Integrar equipos de trabajo en donde se compartan conocimientos y experiencias académicas y laborales.
- Discutir en grupo la información generada por los equipos de trabajo
- Propiciar el uso de las diferentes fuentes de información, tanto de índole primaria como secundaria.

- Elaboración de rúbricas.

9.- SUGERENCIAS DE EVALUACIÓN

La evaluación debe ser continua y cotidiana por lo que se debe considerar el desempeño en cada una de las actividades de aprendizaje, haciendo especial énfasis en:

- Información recabada durante las consultas e investigaciones solicitadas, plasmada en documentos escritos también llamados productos.
- Reportes escritos de los resultados u observaciones obtenidas durante las actividades realizadas en cada unidad académica, así como de las conclusiones obtenidas de dichas observaciones.
- Descripción de otras experiencias concretas que se obtendrán al participar en discusiones, exposiciones o cualquier otro medio didáctico-profesional que trate sobre la materia y que deberán realizarse durante el curso académico.
- Exámenes teórico-prácticos para comprobar la efectividad del estudiante en la resolución de casos prácticos.
- Presentación y exposición de cada actividad de aprendizaje así como de las prácticas propuestas. Algunas se evaluarán por equipo.
- Integración de las rúbricas en un portafolio de evidencias.
- Proyectos realizados empleando cada uno de los modelos estudiados de computación distribuida.

10.- UNIDADES DE APRENDIZAJE

Unidad 1: Introducción a la programación cliente/servidor

Competencias específica a desarrollar	Actividades de aprendizaje
Conocer los conceptos teóricos básicos relacionados con el modelo cliente/servidor.	• Redactar un ensayo de máximo 6 cuartillas sobre el modelo cliente/servidor, especificando las principales arquitecturas lógicas, sus ventajas y desventajas.
Unidad 2: Programación Cliente-Servidor de Bajo Nivel: sockets y canales	
Competencias específica a desarrollar	Actividades de aprendizaje
Identificar las características de la interfaz socket. Utilizar sockets en el desarrollo de aplicaciones básicas cliente/servidor.	• Práctica de ejercicios. Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones en una red de computadoras: • Aplicaciones cliente. • Aplicaciones servidor.
Unidad 3: RMI (Remote Method Invocation, Invocación Remota de Métodos)	

Competencias específica a desarrollar	Actividades de aprendizaje
Identificar las características, ventajas y desventajas del mecanismo RMI de Java para la intercomunicación de aplicaciones mediante la invocación de métodos remotos. Desarrollar aplicaciones empleando el mecanismo RMI.	• Práctica de ejercicios. Desarrollar programas cliente-servidor utilizando Remote Method Invocation (RMI) como tecnología de base, e incluyendo serialización de objetos, control de políticas de seguridad y generación automática de resguardos.
Unidad 4: COM/DCOM (Component Object Model/ Distributed COM)	
Competencias específica a desarrollar	Actividades de aprendizaje
Identificar las características, ventajas y desventajas del Modelo de Objetos de Componentes Distribuidos de Microsoft. Desarrollar aplicaciones bajo DCOM.	• Práctica de ejercicios. Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model, DCOM), utilizando un lenguaje visual (Visual Basic o Visual C#).
Unidad 5: Servicios web XML	
Competencias específica a desarrollar	Actividades de aprendizaje
Comprender el funcionamiento de un servicio web. Desarrollar librerías de métodos remotos para realizar el intercambio de información estructurada en aplicaciones cliente-servidor. Publicar un servicio web para permitir su utilización por aplicaciones cliente.	• Práctica de ejercicios. Desarrollar servicios web y publicarlos en un servidor para invocarlos desde aplicaciones cliente

11.- FUENTES DE INFORMACIÓN

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute , Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

12.- PRÁCTICAS PROPUESTAS

Proyecto. Desarrolle una aplicación chat que le permita comunicarse a los usuarios de una red, tomando en cuenta las siguientes consideraciones:

- En cada cliente de la red deberá instalarse el cliente chat
  - El servidor de chat deberá realizar lo siguiente:
    - ✓ controlar el acceso de los usuarios solicitando su identificación
    - ✓ recibir los mensajes y distribuirlos en la terminal donde se encuentren los usuarios del chat
    - ✓ controlar la no redistribución de mensajes
    - ✓ gestionar la desconexión de un cliente del chat
    - ✓ ocultar la presencia de un cliente a otros clientes del chat
  - Presente sus resultados en la rúbrica de reporte de proyecto final y exponga sus resultados en plenaria.
- Proyecto. Desarrolle un servicio web en donde se implementen métodos de acceso a datos. Posteriormente, el servicio web deberá ser publicado en un servidor a nivel intranet.
- Proyecto. Desarrolle una aplicación cliente en donde se haga el consumo de datos a través del servicio web desarrollado en el punto anterior.
- Proyecto. Desarrolle una aplicación que, mediante RMI, obtenga información del sistema de archivos de una PC remota y genere reportes.

5.- CONTENIDO

5.1. PRÁCTICAS A DESARROLLAR

No. De practica	Nombre	Objetivo	Tema del Programa
1	Configurar o activar los servicios Cliente servidor con ISS y apache:	Contar con un servidor de páginas web para Windows utilizando apache y ISS y comprender genéricamente su funcionamiento en la arquitectura cliente servidor.	1.1. Arquitectura del modelo cliente/servidor
2	Programación concurrente y semáforos en Java.	Desarrollar aplicaciones concurrentes en java dominando los conceptos de sincronía y semáforos.	1.4. Comunicación entre programas
3	Cajera	Desarrollar una aplicación con hilos en java para comprender los conceptos de multitareas	1.4. Comunicación entre programas
4	Servidor de Refranes	Esta práctica consta de dos programas Java ejecutándose de forma independiente RServidor y RCliente. Sin embargo, está constituido por cinco Archivos:	2.4. Desarrollo del lado del servidor con sockets
5	Programa del lado del servidor con sockets	Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones en una red de computadoras	2.4. Desarrollo del lado del servidor con sockets..
6	Aplicaciones Cliente	Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones	2.5. Desarrollo del lado del cliente con sockets..
7	Chat	Desarrollar una aplicación de un chat	2.5. Desarrollo del lado del cliente con sockets
8	RMI	Desarrollar programas cliente-servidor utilizando Remote Method Invocation (RMI) como tecnología de base, e incluyendo serialización de objetos, control de políticas de seguridad y generación automática de resguardos	3.3. Jerarquía de objetos RMI. 3.4. El Sistema de Nombrado Registry.
9	Creación de Servidores COM.	Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model, DCOM), utilizando un lenguaje visual.	4.1. Creación de Servidores COM.

10	Creación de un Cliente COM	Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model, DCOM), utilizando un lenguaje visual.	4.1 Creación de un cliente COM
11	Crear un documento xml para el siguiente	Pedido para el señor Juan Delgado Martínez. El pedido se compone de una bicicleta A2023. A entregar en la calle Reforma # 44, en la fecha 19-11-2018	5.1. Características del lenguaje
12	Servicio Web	Desarrollar servicios web y publicarlos en un servidor para invocarlos desde aplicaciones cliente.	5.2. Visión general de servicios web XML

5.2.- DESARROLLO DE LAS PRÁCTICAS.

De acuerdo al oficio de autorización del año sabático cada una de las prácticas se desarrollará de acuerdo a los siguientes puntos.

I. Número de práctica
II. Nombre
III. Competencia(s) a desarrollar.
IV. Introducción
V. Especificar la correlación con el o los temas y subtemas del programa de estudio vigente. Aplicación en el contexto.
VI. Medidas de seguridad e higiene
VII. Material y equipo necesario
VIII. Metodología
IX. Sugerencias didácticas.
X. Reporte del alumno (discusión de resultados y conclusiones).
XI. Bibliografía (emplear formato APA)

I.- No. De la Práctica.

PRACTICA 1.

II.- Nombre de la Práctica.

Configurar o activar los servicios Cliente servidor con ISS y apache:

III.- Competencia(s) a desarrollar:

- El Conocer los conceptos teóricos básicos relacionados con el modelo cliente/servidor.
- Conocer los distintos aspectos de programación cliente servidor.
- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.

IV.- Introducción.

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente es renderizado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI. El término también se emplea para referirse al computador.

Existen varios servidores web, pero en esta práctica únicamente nos referiremos a los servidores web Apache y ISS que son de los mas utilizados.

SERVIDOR WEB APACHE

EL SERVIDOR HTTP APACHE es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache es altamente configurable, admite bases de datos de autenticación y negociado de contenido, aunque carece de una interfaz gráfica que ayude en su configuración.

CARACTERISTICAS DE APACHE

- Corre sobre Windows y Linux.
- Permite utilizar PHP.
- Permite correr aplicaciones web de .NET (.aspx o .asp) instalando el módulo Mono (no de Microsoft).
- Utilizado por la gran mayoría de hostings comerciales bajo un sistema operativo Linux.

- Es software libre.
- Difícil configurar.
- Permite configuraciones avanzadas

Ventajas:

- Es modular, extensible y multiplataforma
- Puede conectarse directamente a una base de datos
- Es sencillo encontrarle ayudas y soportes
- Tiene amplia captación en la red
- Estabilidad probada en gran cantidad de diversos proyectos
- Interacción con PHP y MySQL

Desventajas:

- Difícil de Administrar
- Difícil a la hora de configurar
- Su actualización no es regular

ES EL SERVIDOR WEB DE MICROSOFT, *IIS*

Internet Information Server, es el motor que ofrece esta compañía a modo profesional, con él es posible programar en ASP (Active Server Pages, Páginas de Servidor Activo) las cuales vienen a ser algo similares al PHP, este servidor posee componentes programables desde ASP accediendo a cada uno de sus módulos para una función específica.

CARACTERISTICAS DE ISS

- Corre sobre Windows.
- Permite utilizar PHP.
- Permite correr aplicaciones web de .NET de forma nativa.
- No tantos proveedores de hosting lo soportan por necesitar un sistema operativo Windows.
- Necesitas una licencia de Windows.
- Fácil configuración.
- Limitación en configuraciones avanzadas.

Ventajas:

- Es confiable, seguro y fácil de utilizar
- No tiene coste adicional, ya que viene incluido en el paquete Windows
- Desarrollada por Microsoft, dándole un respaldo importante
- Se puede administrar a través de Internet
- Soporte ODBC integrado

- Soporta .NET framework y lenguaje ASPX.

Desventajas:

- Su licencia no es gratuita
- No tiene multiplataforma
- Resulta difícil de controlar la dirección
- Código fuente propietario

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

1.- Contexto de la programación cliente/servidor

Subtemas:

1.1. Arquitectura del modelo cliente/servidor.
1.2. Modelos de dos y tres capas.
1.3. Usos y aplicaciones

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.

XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red de Computadoras
- Software de Apache y ISS
- Manual de configuración de Apache y IIS

VIII.- Metodología.

En la práctica Apache se instala de varias formas:

- El programa exclusivo, al que después se le agregan manualmente distintos módulos. Es lo que se explica en este artículo.
- El paquete XAMPP que incluye Apache, la base de datos MySQL, PHP y Perl.
- Otras distribuciones como LAMP o MAMP.

La instalación de la aplicación es sencilla. Descarga de Apache.org. la última versión para Windows, puedes utilizar el siguiente vinculo: ***Descargar Apache*** Crea dos carpetas en la unidad C, la primera de nombre *Apache* y la segunda *servidor_web*. Descomprime el archivo descargado y ejecútalo, sigue los pasos de la instalación y de los datos que te piden solo escoge el destino de la instalación, que será la carpeta que creaste en C:\Apache, los otros datos déjalos de la forma predeterminada para configurarlos más tarde.

El programa al instalarse crea un icono en el área de notificación que te permitirá: iniciar, detener y reiniciar Apache; tienes que tener en cuenta que cualquier cambio que hagas en el archivo de configuración no tendrá efecto hasta que reinicies el servidor.

¿Cómo configurar el Servidor Apache?

Toda la configuración para el funcionamiento de Apache se guarda en un archivo de texto nombrado: *httpd.conf* que se encuentra en la ruta *C:\Apache\conf*, lo podemos editar en cualquier editor de texto como el Bloc de notas pero un programa recomendado es*Notepad*++, software libre que es inmejorable.

La otra opción, más avanzada pero no difícil para configurar apche es la siguiente:

Abre el archivo ***httpd.conf*** y edita manualmente las líneas que se indican a continuación:

Todas las líneas que comienzan con el símbolo # son comentarios, explican en cada sección las distintas opciones, pero se encuentran en inglés.

La línea 52 (Puede variar dependiendo de la versión de apache) **Listen** indica el puerto y dirección IP por el que el servidor va a recibir las peticiones, puedes usarla de las siguientes maneras:

- El servidor va recibir peticiones solo de la misma PC: **Listen localhost:80**
- Recibirá peticiones de otras máquinas en una red local: **Listen 80**
- **Se puede configurar otro puerto 8080**

En la línea 149 **DocumentRoot** es necesario especificar la ruta de la carpeta local que contendrá las páginas y archivos a servir, en tu caso será la carpeta que creaste en *C:/servidor_web*, quedaría de la siguiente forma: **DocumentRoot "C:/servidor_web"**

La línea 177 **<Directory>** establece los permisos necesarios al directorio anterior, quedaría:

<Directory "C:/servidor_web">

Esta es la configuración con los parámetros esenciales para comenzar a utilizar Apache. Guarda los cambios realizados y reinicia el servidor dando clic en el icono del área de notificación.

Comenzar a utilizar Apache

Copia en la carpeta *C:/servidor_web* alguna página web o cualquier archivo y accede a él escribiendo en la barra de direcciones de tu navegador **127.0.0.1** o **localhost**. En el caso de que sea una página web que

estas diseñando la verás cómo realmente se mostrará en Internet. Cualquier problema del servidor estará reflejado en los logs que guarda en*C:\Apache\logs*, los puedes abrir con el bloc de notas, en el archivo access.log se registran todos los accesos hechos al servidor, tanto de tu PC como de internet, en error.log se registran todos los errores de su funcionamiento, te ayudará mucho analizarlos periódicamente.

Si no utilizas el puerto 80 te conectaras como ***localhost:8080***

Acceder al servidor Apache desde la red local

Para acceder al servidor desde otra computadora conectada en una red local solo es necesario escribir en la barra de direcciones la dirección IP de la computadora que sirve de host, es decir la que tiene el servidor Apache instalado.

Para conocer la dirección IP necesaria solo abre una ventana de CMD, escribiendo en Inicio >CMD, escribe en la ventana que se abre IPCONFIG /ALL y oprime Enter, busca la línea *Puerta de enlace*, el número a continuación es la dirección IP del proxy.

En caso de que el servidor escuche en otro puerto diferente al 80 (predeterminado) escribe: "dirección ip:puerto", por ejemplo: 192.168.1.3:8080

En caso de conflictos al tratar de conectarse a un equipo usando una red local, verifica lo siguiente:

- La dirección IP del equipo al que deseas conectarte.
- Si Apache está escuchando en el puerto al que se efectúa la petición.
- Si el firewall de Windows está bloqueando la conexión.
- Si se recibe un mensaje de error con el código 403 significa que no se cuenta con los permisos necesarios para acceder al directorio, en ese caso establécelo de la siguiente forma:

```
<Directory " ruta al directorio">
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

Como conectarse a internet a través de Apache desde tu PC

Para que tu conexión a internet pase a través del servidor será necesario configurarlo como un proxy fordward para eso en el archivo de configuración httpd.conf descomenta, (quitar el signo #) las siguientes líneas:

- LoadModule proxy_module modules/mod_proxy.so

- LoadModule proxy_http_module modules/mod_proxy_http.so

Después agrega en el final del archivo la siguiente linea: **ProxyRequests On** Cierra y guarda los cambios.

- Accede a las *Opciones de internet* mediante el Panel de Control.
- En la pestaña *Conexiones* pulsa el botón *Configuración de LAN* y marca la casilla*Usar un servidor Proxy*
- Escribe en *Dirección:* ***127.0.0.1*** y en *Puerto:* ***80*** o el que vayas a usar.
- Presiona *Aceptar* en todas las ventanas.
- Reinicia el servidor.

Lo anterior se aplica si usas el navegador Internet Explorer y Google Chrome, si usas Firefox las opciones anteriores tienes que ingresarlas en: *Opciones >Configuración >Configurar como Firefox se conecta a Internet*.

A partir de ahora toda tu conexión pasa a través de Apache, sea direcciones locales o externas.

Como guardar en la cache del servidor web las páginas

Para guardar en la cache del servidor web las páginas web haz lo siguiente: Crea una carpeta en la unidad C de nombre *cacheroot*, será la que almacene los archivos de la cache, también puedes crearla en otra unidad, en ese caso tendrías que indicar su ruta en el archivo de configuración.

En el archivo httpd.conf descomenta, (quita el signo #) la siguiente línea:

LoadModule expires_module modules/mod_expires.so

Y agrega las siguientes líneas:

LoadModule cache_module modules/mod_cache.so

LoadModule disk_cache_module modules/mod_disk_cache.*so*

Copia y pega las siguientes líneas de código al final del archivo:

```
<IfModule mod_disk_cache.c>
CacheRoot c:/cacheroot
#CacheEnable disk /
CacheDirLevels 5
CacheDirLength 3
CacheDefaultExpire 86400
CacheIgnoreNoLastMod On
CacheStoreNoStore On
```

```
CacheStorePrivate On
CacheEnable disk http://*

</IfModule>

<Directory "C:\cacheroot">
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>

<IfModule mod_expires.c>
ExpiresActive on
ExpiresDefault A604800
</IfModule>
```

El valor *ExpiresDefault A604800* especifica el tiempo en segundos que persistirá tu cache, puedes modificarlo de acuerdo a tus necesidades. Todos los demás valores puedes ajustarlo a tu conveniencia.

Lee las siguientes páginas en la carpeta de manuales:

C:/Apache/manual/mod/mod_disk_cache.html
C:/Apache/manual/mod/mod_expires.html

Reinicia el servidor.

Manual de Apache offline

La instalación de Apache incluye un manual bastante completo en varios idiomas.

Para poder acceder con el navegador a todas las paginas haz lo siguiente:

Abre el archivo de configuración de Apache "httpd.conf" y descomenta (quita la almohadilla) la siguiente línea:

include conf/extra/httpd-manual.conf"

Ahora solo es necesario usar la siguiente dirección en el navegador web para cargar los archivos del manual:

http://sistemas.ith.mx/manual/

Sustituye sistemas.ith.mx por el nombre de tu sitio.

Instalar IIS en Windows

Si hemos visto las principales alternativas del mercado y hemos decidido optar por instalar IIS en un Windows, deberíamos de saber algo: IIS no se instala, se habilita.

Es decir***, IIS*** es un complemento a Windows, por lo que no tendremos que bajar ninguna aplicación ni nada, simplemente tendremos que seguir unos pasos para tenerlo instalado y habilitado.

1. ***Ves a Panal de control***
2. ***Ves a "Programas"***
3. ***Dependiendo la vista que tengas podrás darle a "Activar o desactivar las características de Windows", si no ves esa opción presiona sobre "Programas y características".***
4. ***A la izquierda presiona sobre "Activar o desactivar las características de Windows".***

Si has hecho todo esto te saldrá las siguientes opciones:

En este caso, sólo tendrás que seleccionar IIS (Internet Information Sever) y presionar en "aceptar", puedes seleccionar todas las opciones dentro de IIS para que se te habilite e instale, no obstante, si no sabes muy bien si instalarlo y tienes disco duro, te recomiendo instalarlo todo (a no ser, por ejemplo, que ya tengas un servidor FTP).

Una vez instalado puedes probar al poner la IP local (127.0.0.1) o directamente "localhost" en el navegador y ya deberías de tener acceso.

Si deseas tener acceso desde otra computadora de la red local tendrás que poner su la dirección ip de la maquina donde habilitaste ***IIS***

Como usar el servidor Apache y IIS en el mismo equipo

Es imposible utilizar dos servidores web al mismo tiempo en el equipo, pero si te interesa instalar o ya usas en tu PC el servidor web que incluye Windows, *Internet Information Services (IIS)*, puedes utilizar un sencillo script para alternar el uso de ambos. Es un archivo batch que inicia y detiene los servicios de ambos servidores de forma alterna según se seleccione.

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Compruebe que los puertos de comunicación sean los indicados
- Verifique su funcionamiento desde localhost.
- Verifique su funcionamiento desde otras computadoras.

- Investigar otros servidores Web (tomcat, etc.)

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el laboratorio de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (formato APA).

1. Scott Hawkins, (2002), Guía *Esencial de Apache,* Prentice Hall.
2. https://www.fdi.ucm.es/profesor/jpavon/web/31-ServidoresWeb-Apache.pdf
3. http://www.juntadeandalucia.es/empleo/recursos/material_didactico/especialidades/materialdidactico_administrador_servidores/Content/3-servicios_web/5-ApacheWebServer.pdf
4. https://help.mproerp.com/help/0000000059/
5. http://www.juntadeandalucia.es/empleo/recursos/material_didactico/especialidades/materialdidactico_administrador_servidores/Content/3-servicios_web/2-IIS.pdf

I.- No. De la Práctica.

PRACTICA 2.

II.- Nombre de la Práctica.

Programación concurrente y semáforos en Java.

III.- Competencia(s) a desarrollar:

- El Conocer los conceptos teóricos básicos relacionados con el modelo cliente/servidor.
- Conocer los distintos aspectos de programación cliente servidor.
- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.
- Desarrollar aplicaciones concurrentes en java dominando los conceptos de sincronía y semáforos.

IV.- Introducción.

Programación concurrente

- Consiste en la ejecución simultánea de varias tareas, que pueden interactuar entre sí.
- Paso de mensajes.
- Acceso a memoria compartida.
- Un hilo (thread) es una tarea (conjunto de instrucciones) que puede ejecutarse en paralelo con las demás.
- Estos hilos pueden ser todos ejecutados por un solo procesador, o distribuidos entre los distintos procesadores.
- La asignación de hilos a procesadores es realizada por el sistema operativo.

En Java es posible ejecutar tareas en paralelo, utilizando hebras de control (hilos, threads). Este modo de programación permite tener un espacio de memoria, código o recursos compartidos. Tiene la ventaja que su ejecución resulta más económica que un proceso completo.

Semáforos

• Java cuenta con semáforos implícitos de la forma:

```
Object mutex = new Object();
/*
…*/
Synchonized (mutex){
/*
…
*/
```

}

Que solo pueden ser utilizados para exclusión mutua.
Solo una hebra de control puede ejecutarse en el bloque synchonized en un momento dado.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

1. Contexto de la programación cliente/servidor

Subtemas:

1.3. Usos y aplicaciones
1.4. Comunicación entre programas

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a)

del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.

XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.

XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red local de computadoras
- Apuntes sobre sincronía y semáforos en java

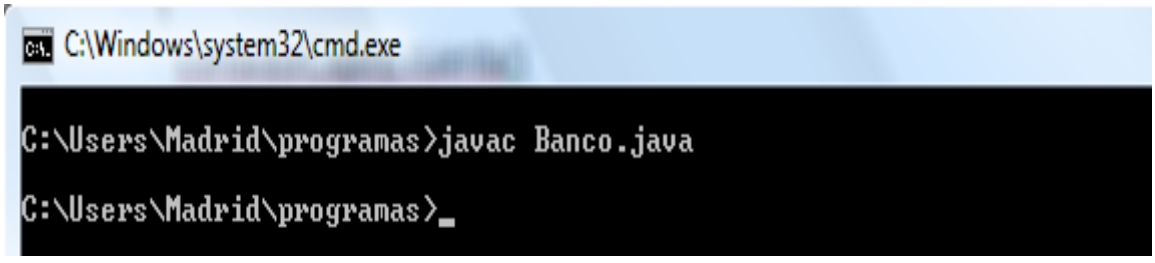
VIII.- Metodología.

Utilizando un editor de texto planto escribiremos el siguiente código y lo guardaremos como Banco.java

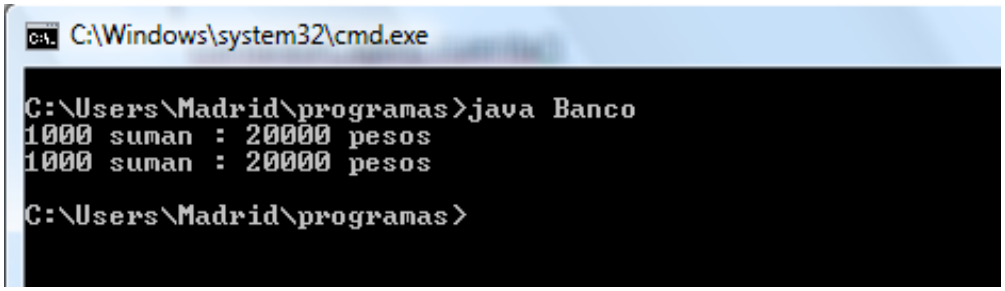
```
import java.io.*;
class Banco
{
  public static void main(String args[])
   {
     try
       {
       //Declaramos los dos montones de billetes
       Contador co1 = new Contador();
       Contador co2 = new Contador();

       //Declaramos los dos cajeros
       Cajero c1 = new Cajero(co1);
       Cajero c2 = new Cajero(co2);

       // Se ponen a contar..
       c1.start();
       c2.start();
       c1.join();
       c2.join();
       }
     catch (Exception e){
       e.printStackTrace();
       }
    }
}
```

```
class Contador
{
  int numBilletes = 0;
  long suma = 0;
  final int TOTAL_BILLETES = 1000;
  final int VALOR_BILLETES = 20;
  void cuenta()
   {
   //A contar la suma de los billetes
   for(numBilletes = 0; numBilletes < TOTAL_BILLETES; numBilletes ++)
     {
      suma += VALOR_BILLETES;

      //Billetes de 200 pesos
      Thread.yield();
      }
   System.out.println(numBilletes + " suman : " + suma + " pesos");
   }
}

class Cajero extends Thread
{
  Contador contadorCajero;
  Cajero(Contador paramContador)
   {
    contadorCajero = paramContador;
   }

   public void run()
   {
    contadorCajero.cuenta();
   }
}
```

Compilamos el programa. Banco.java



Ejecutamos el programa. Banco



Compartiendo el recurso

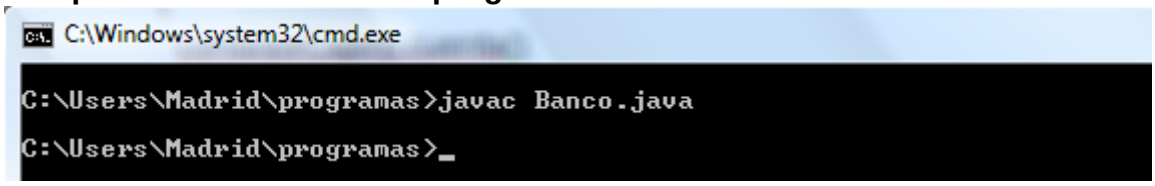
- ✓ Ahora supongamos que los dos cajeros deben contar del mismo montón, o sea, lo comparten y por tanto, la suma de lo que haya contado cada uno debe ser el resultado total.
- ✓ Para ello, modificaremos el código añadiendo lo siguiente

Modificamos el programa anterior.

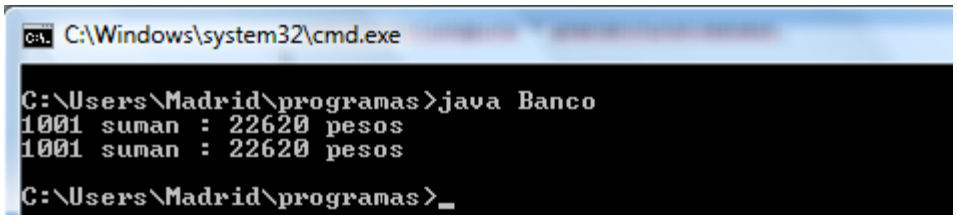
Teníamos declaramos los dos montones de billetes:

```
Contador co1 = new Contador();
// Sobra Contador co2 = new Contador();

//Declaramos los dos cajeros al mismo montón.
Cajero c1 = new Cajero(co1);
Cajero c2 = new Cajero(co1);
```

Compilamos nuevamente el programa.



Ejecutamos nuevamente el programa.



1001 suman: 22620 pesos
1001 suman: 22620 pesos

Este Resultado es Incorrecto.

- ✓ El resultado anterior es incorrecto.
- ✓ Por tanto, debemos utilizar un mecanismo de sincronización que garantice que cuando un cajero cuente un billete y lo sume, el otro no pueda intentar coger el mismo billete y sumarlo.
- ✓ La solución que ofrece Java para resolver este problema es de lo más simple y eficiente, utilizando la cláusula synchronized en la declaración del método donde se realiza la tarea "crítica".
- ✓ Por tanto, cambiaremos el método void cuenta() por:

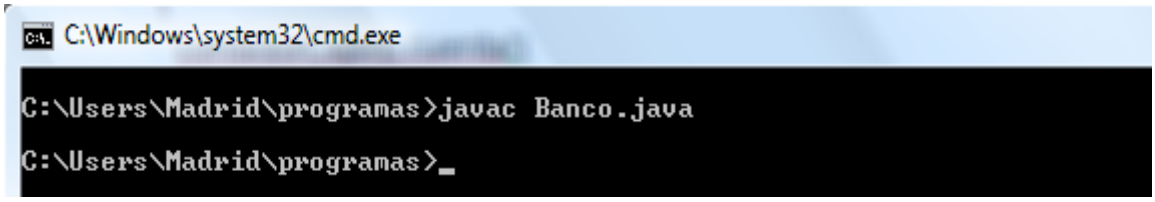
synchronized void cuenta()

Modificamos el Programa.

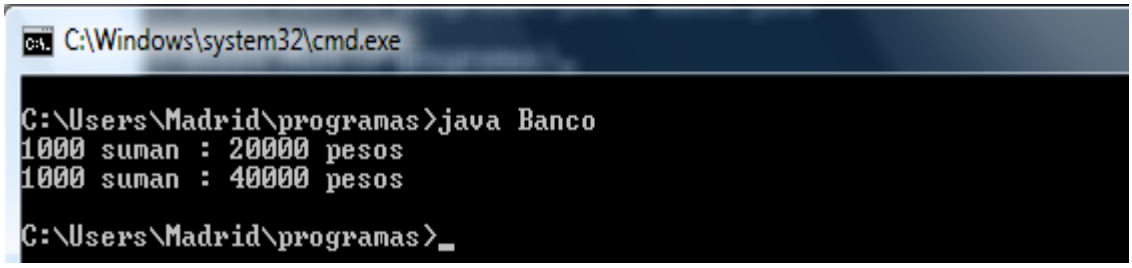
El método void cuenta ahora es:

synchronized void cuenta()

Compilamos el programa nuevamente.



Ejecutamos el programa.



• Si realizamos ese cambio, obtenemos el siguiente resultado:

- 1000 suman: 20000 pesos
- 1000 suman: 40000 pesos

• Esto ocurre porque no se inicializa la variable suma antes del ciclo que cuenta los billetes, por lo que el segundo cajero continúa la suma en donde la dejó el anterior.

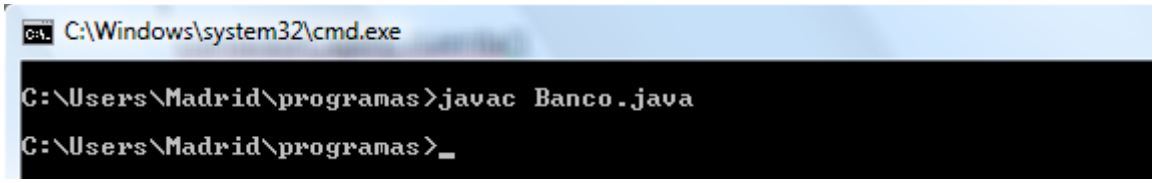
Sigue siendo incorrecto.

Seguimos Modificando el programa.

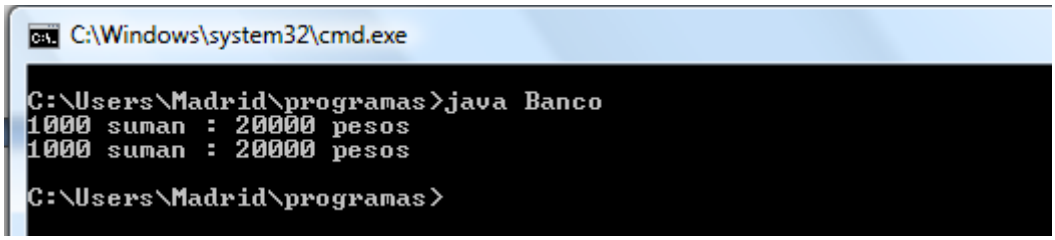
Si modificamos el código e incluimos la inicialización, tendremos:

```
void cuenta()
   {
    //Cada cajero cuenta lo suyo…
   suma = 0;

   //A contar la suma de los billetes
  for(numBilletes = 0; numBilletes < TOTAL_BILLETES; numBilletes++)
    {
      suma += VALOR_BILLETES;
     Thread.yield();
    }
}
```

Compilamos nuevamente el programa.



Ejecutamos el programa.



Al fin, resultado correcto.

Realice un ejercicio con tres contadores de billetes, obviamente, utilizando 3 cajeros.

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique el algoritmo
- Codifique el programa
- Ejecute el programa
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el laboratorio de clase y esta deberá subirse a la página de cursos en línea del Tecnológico en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (formato APA).

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
5. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
6. Froute , Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 3.

II.- Nombre de la Práctica.

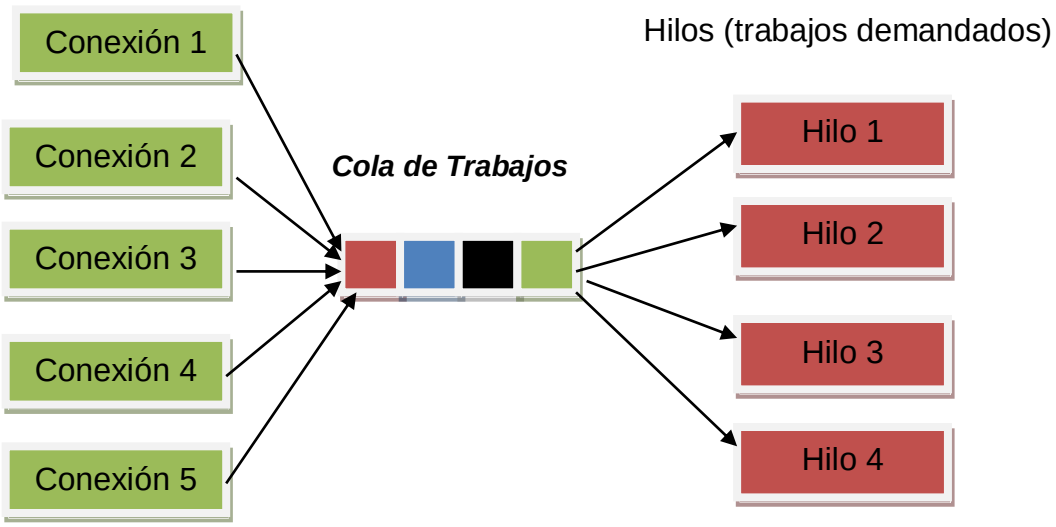
Cajera

III.- Competencia(s) a desarrollar:

- El Conocer los conceptos teóricos básicos relacionados con el modelo cliente/servidor.
- Conocer los distintos aspectos de programación cliente servidor.
- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.
- Desarrollar una aplicación con hilos en java para comprender los conceptos de multitareas

IV.- Introducción.

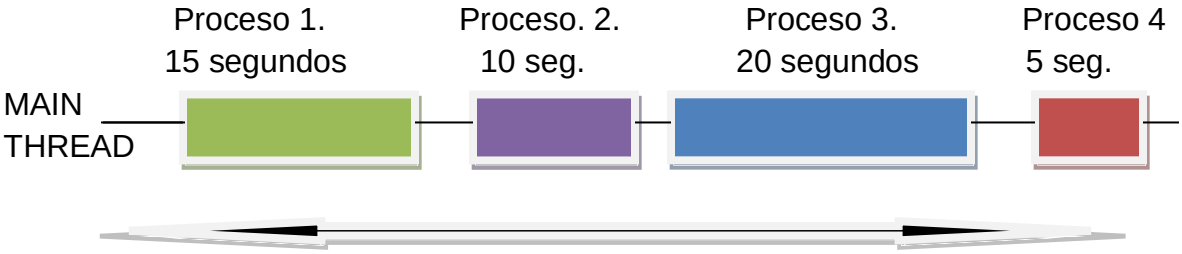
Hilos en java.



En este ejercicio vamos a ver otra manera de cómo trabajar con Threads en Java (o hilos en español).

En esencia la multitarea nos permite ejecutar varios procesos a la vez; es decir, de forma concurrente y por tanto eso nos permite hacer programas que se ejecuten en menor tiempo y sean más eficientes. Evidentemente no podemos ejecutar infinitos procesos de forma concurrente ya que el hardware tiene sus limitaciones, pero raro es al día de hoy la computadora que no tengan más de un núcleo por

tanto en un procesador con dos núcleos se podrían ejecutar dos procesos a la vez y así nuestro programa utilizaría al máximo los recursos hardware. Para que veas la diferencia en un par de imágenes, supongamos que tenemos un programa secuencial en el que se han de ejecutar 4 procesos; uno detrás de otro, y estos tardan unos segundos:



El programa tarda en ejecutarse 50 segundos (15 + 10 + 20 + 5)

Si en vez de hacerlo de forma secuencial, lo hiciésemos con 4 hilos, el programa tardaría en ejecutarse solo 20 segundos, es decir el tiempo que tardaría en ejecutarse el proceso más largo. Esto evidentemente sería lo ideal, pero la realidad es que no todo se puede paralelizar y hay que saber el número de procesos en paralelo que podemos lanzar de forma eficiente. En principio en este ejercicio no vamos a hablar sobre ello ya que el objetivo de la misma es ver cómo se utilizan los hilos en java con un ejemplo relativamente sencillo y didáctico.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

1. Contexto de la programación cliente/servidor

Subtemas:

1.3. Usos y aplicaciones
1.4. Comunicación entre programas

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.

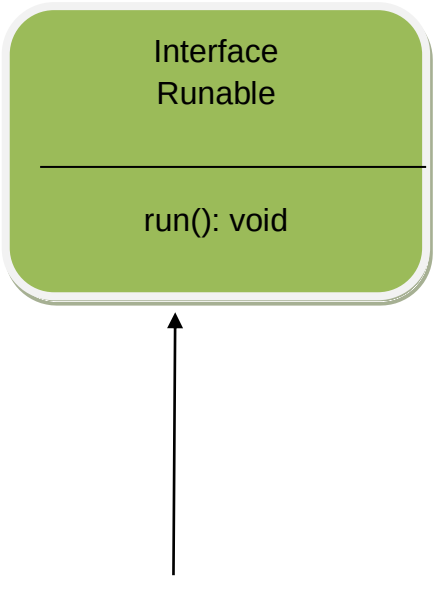
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

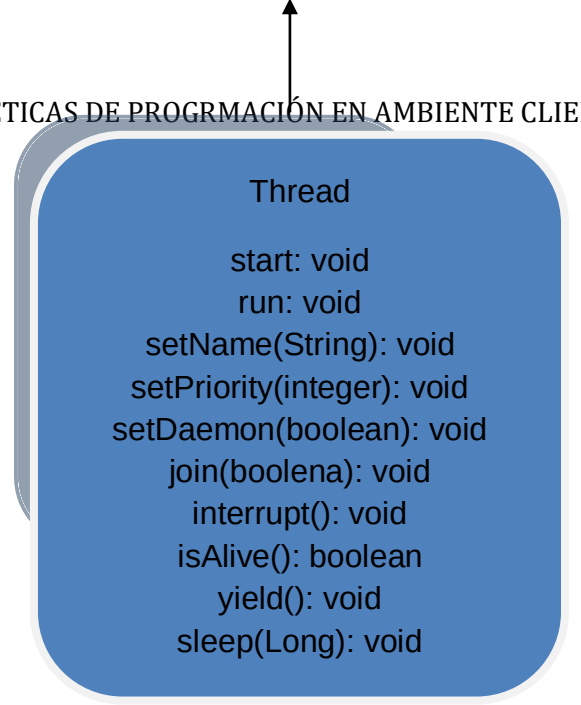
VII.- Material y equipo necesario.

- Computadora.
- Red local de computadoras
- Apuntes de Java.

VIII.- Metodología.

En Java para utilizar la multitarea debemos de usar la clase Thread (es decir que la clase que implementemos debe heredar de la clase Thread) y la clase Thread implementa la Interface Runnable. En el siguiente diagrama de clase mostramos la Interface Runnable y la clase Thread con sus principales métodos:





En esta práctica no vamos a ver cómo utilizar todos los métodos de la clase Thread, vamos a poner un ejemplo para que analices las ventajas de la multitarea, viendo cómo se ejecutaría un programa sin utilizar la multitarea y otro utilizándola.

En esta práctica vamos a simular el proceso de cobro de un supermercado; es decir, unos clientes van con un carro lleno de productos y una cajera les cobra los productos, pasándolos uno a uno por el escáner de la caja registradora. En este caso la cajera debe de procesar la compra cliente a cliente, es decir que primero le cobra al cliente 1, luego al cliente 2 y así sucesivamente. Para ello vamos a definir una clase “Cajera” y una clase “Cliente” el cual tendrá un “array de enteros” que representaran los productos que ha comprado y el tiempo que la cajera tardará en pasar el producto por el escáner; es decir, que si tenemos un array con [1,3,5] significará que el cliente ha comprado 3 productos y que la cajera tardara en procesar el producto 1 ‘1 segundo’, el producto 2 ‘3 segundos’ y el producto 3 en ‘5 segundos’, con lo cual tardara en cobrar al cliente toda su compra ‘9 segundos’.

Explicado esta práctica vamos a ver cómo hemos definido estas clases:

Clase ***Cajera.java***

```
public class Cajera {

        private String nombre;
        public Cajera() {
        }

        public Cajera(String nombre) {
                this.nombre = nombre;
        }
        public String getNombre() {
```

```
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;

	}

	public void procesarCompra(Cliente cliente, long timeStamp) {

		System.out.println("La cajera " + this.nombre +
				" COMIENZA A PROCESAR LA COMPRA DEL
CLIENTE " + cliente.getNombre() +
				" EN EL TIEMPO: " + (System.currentTimeMillis() -
timeStamp) / 1000 +
				"seg");

		for (int i = 0; i < cliente.getCarroCompra().length; i++) {

	this.esperarXsegundos(cliente.getCarroCompra()[i]);
				System.out.println("Procesado el producto " + (i + 1)
+
				" ->Tiempo: " + (System.currentTimeMillis() -
timeStamp) / 1000 +
				"seg");
		}

		System.out.println("La cajera " + this.nombre + " HA
TERMINADO DE PROCESAR " +
				cliente.getNombre() + " EN EL TIEMPO: " +
				(System.currentTimeMillis() - timeStamp) / 1000 +
"seg");

	}

private void esperarXsegundos(int segundos) {
	try {
		Thread.sleep(segundos * 1000);
	} catch (InterruptedException ex) {
		Thread.currentThread().interrupt();
	}
   }

}
```

Clase ***Cliente.java***

```
public class Cliente {

	private String nombre;
	private int[] carroCompra;

	public Cliente() {
	  }
	public Cliente(String nombre, int[] carroCompra) {
		this.nombre = nombre;
		this.carroCompra = carroCompra;
	}

	public String getNombre() {
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public int[] getCarroCompra() {
		return carroCompra;
	}

	public void setCarroCompra(int[] carroCompra) {
		this.carroCompra = carroCompra;
	}

}
```

Si ejecutásemos este programa propuesto con dos Clientes y con un solo proceso (que es lo que se suele hacer normalmente), se procesaría primero la compra del Cliente 1 y después la del Cliente 2, con lo cual se tardará el tiempo del Cliente 1 + Cliente 2. A continuación vamos a ver como programamos el método Main para lanzar el programa. CUIDADO: Aunque hayamos puesto dos objetos de la clase Cajera (cajera1 y cajera2) no significa que tengamos dos cajeras independientes, lo que estamos diciendo es que dentro del mismo hilo se ejecute primero los métodos de la cajera1 y después los métodos de la cajera2, por tanto, a nivel de procesamiento es como si tuviésemos una sola cajera:

Clase ***Main.java***

```
public class Main {
	public static void main(String[] args) {
```

```
            Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2,
3 });
            Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1
});

            Cajera cajera1 = new Cajera("Cajera 1");
            Cajera cajera2 = new Cajera("Cajera 2");

            // Tiempo inicial de referencia
            long initialTime = System.currentTimeMillis();

            cajera1.procesarCompra(cliente1, initialTime);
            cajera2.procesarCompra(cliente2, initialTime);
        }
}
```

Como vemos se procesa primero la compra del cliente 1 y después la compra del cliente 2 tardando en procesar ambas compras un tiempo de 26 segundos.

¿Y si en vez de procesar primero un cliente y después otro, procesásemos los dos a la vez?, ¿Cuánto tardaría el programa en ejecutarse? Pues bien, si en vez de haber solo una Cajera (es decir un solo hilo), hubiese dos Cajeras (es decir dos hilos o threads) podríamos procesar los dos clientes a la vez y tardar menos tiempo en ejecutarse el programa. Para ello debemos de modificar la clase "Cajera.java" y hacer que esta clase herede de la clase Thread para heredar y sobre-escribir algunos de sus métodos. Primero vamos a ver como codificamos esta nueva clase "CajeraThread.java" y después explicamos sus características.

Clase ***CajeraThread.java***

```
public class CajeraThread extends Thread {

        private String nombre;
        private Cliente cliente;
        private long initialTime;
        private Thread thread;
        public CajeraThread() {

        }

        public CajeraThread(String nombre, Cliente cliente, long initialTime) {
            this.nombre = nombre;
            this.cliente = cliente;
            this.initialTime = initialTime;
      }
        public String getNombre() {
```

```
		return nombre;
	}

	public void setNombre(String nombre) {
		this.nombre = nombre;
	}

	public long getInitialTime() {
		return initialTime;
	}

	public void setInitialTime(long initialTime) {
		this.initialTime = initialTime;
	}

	public Cliente getCliente() {
		return cliente;
	}

	public void setCliente(Cliente cliente) {
		this.cliente = cliente;
	}



	public void run() {

		System.out.println("La cajera " + this.nombre + " COMIENZA A
PROCESAR LA COMPRA DEL CLIENTE "
					+ this.cliente.getNombre() + " EN EL TIEMPO:
"
					+ (System.currentTimeMillis() -
this.initialTime) / 1000
					+ "seg");

		for (int i = 0; i < this.cliente.getCarroCompra().length; i++) {
			this.esperarXsegundos(cliente.getCarroCompra()[i]);
			System.out.println("Procesado el producto " + (i + 1)
			+ " del cliente " + this.cliente.getNombre() + "->Tiempo: "
			+ (System.currentTimeMillis() - this.initialTime) / 1000
			+ "seg");
		}

		System.out.println("La cajera " + this.nombre + " HA
TERMINADO DE PROCESAR "
						+ this.cliente.getNombre() + " EN EL
TIEMPO: "
```

```
                                        + (System.currentTimeMillis() -
this.initialTime) / 1000
                                        + "seg");
        }

    public void start() {
            thread = new Thread(this);
            thread.start();
        }

        private void esperarXsegundos(int segundos) {
            try {
                Thread.sleep(segundos * 1000);
            } catch (InterruptedException ex) {
                Thread.currentThread().interrupt();
            }
        }

}
```

Lo primero que vemos y que ya hemos comentado es que la clase “CajeraThread” debe de heredar de la clase Thread: “extendsThread“.

Otra cosa importante que vemos es que hemos sobre-escrito el método “run()” (de ahi la etiqueta @Override) . Este método es imprescindibles sobre-escribirlo (ya que es un método que está en la clase Runnable y la clase Thread Implementa esa Interface) porque en él se va a codificar la funcionalidad que se ha de ejecutar en un hilo; es decir, que lo que se programe en el método “run()” se va a ejecutar de forma secuencial en un hilo. En esta clase “CajeraThread” se pueden sobre-escribir más métodos para que hagan acciones sobre el hilo o thread como, por ejemplo, parar el thread, ponerlo en reposos, etc. A continuación, vamos a ver como programamos el método Main para que procese a los clientes de forma paralela y ver como se tarda menos en procesar todo. El método Main está en la clase “MainThread.java” que tiene el siguiente contenido:

Clase ***MainRunnable.java***

```
public class MainRunnable implements Runnable{

        private Cliente cliente;
        private Cajera cajera;
        private long initialTime;

        public MainRunnable (Cliente cliente, Cajera cajera, long initialTime){
            this.cajera = cajera;
            this.cliente = cliente;
            this.initialTime = initialTime;
```

```
	}

	public static void main(String[] args) {

		Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2,
3 });
		Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1
});

		Cajera cajera1 = new Cajera("Cajera 1");
		Cajera cajera2 = new Cajera("Cajera 2");

		// Tiempo inicial de referencia
		long initialTime = System.currentTimeMillis();

		Runnable proceso1 = new MainRunnable(cliente1, cajera1,
initialTime);
		Runnable proceso2 = new MainRunnable(cliente2, cajera2,
initialTime);

		new Thread(proceso1).start();
		new Thread(proceso2).start();
	}

	public void run() {
		this.cajera.procesarCompra(this.cliente, this.initialTime);
	}
}
```

En este ejemplo vemos como el efecto es como si dos cajeras procesasen la compra de los clientes de forma paralela sin que el resultado de la aplicación sufra ninguna variación en su resultado final, que es el de procesar todas las compras de los clientes de forma independiente.

Conclusión.

El concepto de multitarea o multiprocesamiento es bastante sencillo de entender ya que solo consiste en hacer varias cosas a la vez sin que se vea alterado el resultado final. Como ya se ha dicho en la entrada no todo se puede paralelizar y en muchas ocasiones suele ser complicado encontrar la manera de paralelizar procesos dentro de una aplicación sin que esta afecte al resultado de la misma, por tanto, aunque el concepto sea fácil de entender el aplicarlo a un caso práctico puede ser complicado para que el resultado de la aplicación no se vea afectado.

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.

- Identifique el algoritmo
- Codifique el programa
- Ejecute el programa
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía *(formato APA).*

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute , Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 4.

II.- Nombre de la Práctica.

Servidor de Refranes

III.- Competencia(s) a desarrollar:

Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.

- Utilizar sockets en el desarrollo aplicaciones cliente/servidor
- Identificar las características de la interfaz socket.
- Utilizar sockets en el desarrollo de aplicaciones básicas cliente/servidor.

IV.- Introducción.

Las redes actuales utilizan el *packet switching* para la transferencia de datos. Los datos se envuelven en paquetes que se transfieren desde un origen a un destino, donde se extraen de uno en uno los datos de uno o más paquetes para reconstruir el mensaje original.

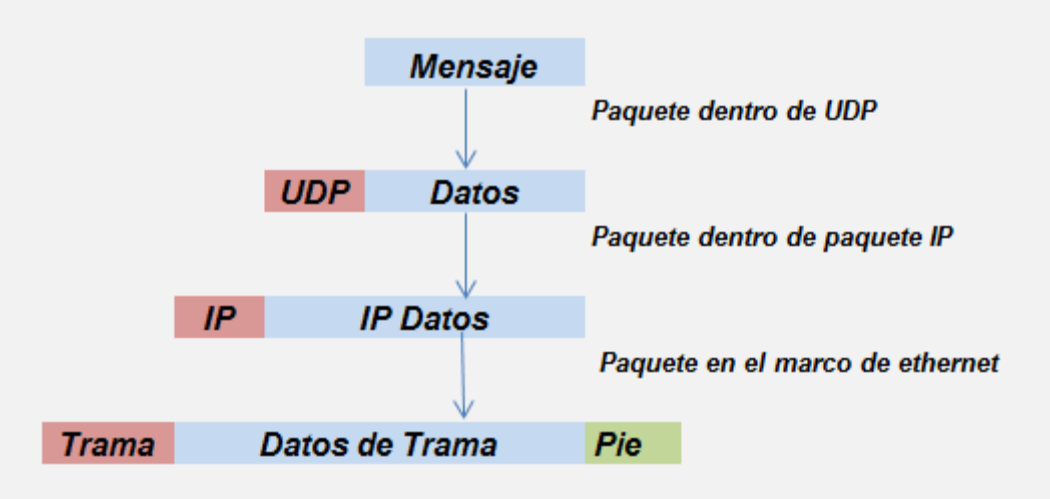
Los nodos que se comunican a través de Internet utilizan principalmente dos protocolos:

- TCP - *Transsmision Control Protocol*
- UDP – (Universal User) Datagram Protocol

El protocolo UDP - (User Universal) Datagram Protocol - se utiliza para comunicaciones en la que no se garantiza una transmisión fiable (reliable). UDP no está orientado a conexión, por lo tanto, no garantiza la entrega. UDP envía paquetes de datos independientes, denominados *datagramas*, desde una aplicación a otra.

Protocolo de datagramas de usuarios	
Familia de protocolos de internet	
Función intercambio de datagramas a través de una Red	
Ubicación en la pila de protocolos	
Aplicación	***DNS, DHCP, NTP, ….***
Transporte	***UDP***
Red	***IP***
Enlace	***Ethernet, Token ring, FDDI***
Estándares	

El envío de datagramas es similar a enviar una carta a través del servicio postal: El orden de salida no es importante y no está garantizado, y cada mensaje es independiente de cualquier otro.



En las comunicaciones basadas en datagramas como las UDP, el paquete de datagramas contiene el número de puerto de su destino y UDP encamina el paquete a la aplicación apropiada, como ilustra la figura 1.

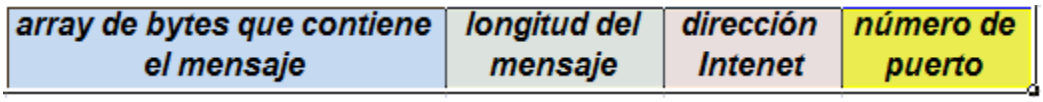
El API Java para UDP proporciona una abstracción del ``paso de mensajes", esto es, la forma más simple de comunicación entre computadoras. Esto hace posible a un proceso emisor transmitir un único mensaje a un proceso receptor. Los paquetes independientes que contienen esos mensajes se denominan datagramas. En Java, el emisor especifica el destino usando un *socket* (una referencia indirecta a un puerto particular usada por el proceso receptor en la máquina receptora).

Un datagrama enviado mediante UDP es trasmitido desde un proceso emisor a un proceso receptor sin reconocimiento o re comprobaciones. Si tiene lugar un fallo, el mensaje puede no llegar. Un datagrama es transmitido entre procesos cuando un proceso lo envía y otro proceso lo recibe. Cualquier proceso que necesite enviar o recibir mensajes debe en primer lugar crear un *socket* a una dirección de Internet y a un puerto local. Un servidor enlazará ese *socket* a un puerto servidor - uno que se hace conocido a los clientes de manera que puedan enviar mensajes al mismo. Un cliente enlaza su *socket* a cualquier puerto local libre. El método receptor devuelve la dirección de Internet y el puerto del emisor, además del mensaje, permitiendo a los receptores enviar una respuesta.

Las clases Java para establecer comunicaciones mediante datagramas son***: DatagramPacket y DatagramSocket***.

La clase *DatagramPacket*

La clase DatagramPacket proporciona un constructor que permite crear instancias de un array de bytes parar: el mensaje, la longitud del mensaje, la dirección Internet y el puerto local del *socket* de destino, de la siguiente forma:



Los objetos del tipo DatagramPacket se pueden transmitir entre procesos cuando un proceso los envía y otro los recibe.

Esta clase proporciona otro constructor para usarlo cuando se recibe un mensaje. Sus argumentos especifican un array de bytes en el que recibir el mensaje y la longitud del array. Cuando se recibe un mensaje se pone en el DatagramPacket junto con su longitud, la dirección de Internet y el puerto del *socket* de envío.

Se puede obtener el mensaje del objeto DatagramPacket mediante el método getData(). Los métodos getPort() y getAddress() permiten obtener el puerto y la dirección Internet del objeto de tipoDatagramPacket.

El proceso receptor del mensaje tiene que especificar un array de bytes de un tamaño determinado en el cual recibir el mensaje, esto es, ha de predecir el *Tamaño del Mensaje*. Si el mensaje es muy grande para el array se trunca cuando llega. El protocolo IP subyacente permite longitudes de paquetes de más de 2^{16} bytes, que incluye tanto las cabeceras como los mensajes. Sin embargo, la mayoría de los entornos imponen una restricción en el tamaño a 8 kilobytes. Cualquier aplicación que necesite mensajes mayores que el máximo, debe fragmentarlos en pedazos de ese tamaño. Generalmente, una aplicación decidirá sobre un tamaño que no sea excesivamente grande pero que se adecue a su uso previsto.

La clase DatagramSocket

La clase DatagramSocket da soporte a *sockets* para el envío y recepción de datagramas UDP.

Se proporciona un constructor que toma un puerto como argumento, para que sea usado por los procesos que necesitan usar un puerto particular. También se proporciona un constructor sin argumentos que permite al sistema escoger un puerto local libre. Estos constructores pueden lanzar una excepción del tipo SocketException si el puerto ya está en uso o si está reservado.

Esta clase cuenta con los siguientes métodos:

- ***send() y receive().***

  Estos métodos permiten transmitir datagramas entre un par de *sockets*. El argumento del send es una instancia de un DatagramPacket que contiene un mensaje y su destino. El argumento del receive es un objeto DatagramPacket vacío en el cual se pondrá el mensaje, su longitud y su origen. Tanto el método send() como el receive() pueden lanzar una IOException.

  Las comunicaciones mediante datagramas de UDP usan envíos no bloqueantes (*non-blocking sends*) y recepciones bloqueantes (*blocking receives*). Las operaciones de envío retornan cuando estas han dado el mensaje a los protocolos IP o UDP subyacentes, los cuales son responsables de trasmitirlos a su destino. En la llegada, el mensaje es puesto en una cola por el *socket* que está asociado al puerto de destino. El mensaje puede ser recogido de la cola por una excepción o llamadas futuras de recepción (receive()) sobre ese *socket*. Los mensajes son descartados en el destino si ningún proceso tiene asociado un *socket* al puerto de destino. El método receptor (receive()) se bloquea hasta que se recibe un datagrama, a menos que se establezca un tiempo límite (timeout) sobre el*socket*. Si el proceso que invoca al método receive() tiene otra tarea que hacer mientras espera por el mensaje, debería planificarse en un flujo de ejecución (*thread*) separado.

- ***setSoTimeout().***

  Este método permite establecer un tiempo de espera. Con un tiempo de espera establecido, el método receive() se bloqueará por el tiempo especificado y entonces lanzará unaInterruptedIOException().

- ***connect().***

  Este método se utiliza para conectar a un puerto remoto particular y una dirección de Internet, en este caso el *socket* sólo es capaz de enviar y recibir mensajes desde esa dirección.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

2. Programación Cliente Servidor de Bajo Nivel: sockets y canales

Subtemas:

2.1. Concepto de socket.
2.2. Dominios y Tipos de sockets.
2.3. Creación/ implementación y supresión de sockets.
2.4. Desarrollo del lado del servidor con sockets.
2.5. Desarrollo del lado del cliente con sockets

VI. Medidas de seguridad e higiene

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red local de computadoras
- Apuntes de java (utilización de las clases de socket en java)

VIII.- Metodología.

La siguiente práctica implementa un servidor de refranes.

El ejemplo consta de dos programas Java ejecutándose de forma independiente RefServidor y RefClientes. Sin embargo, está constituido por cinco Archivos:

- RefServidor.java (implementación del Servidor)
- RefServidorThread.java (implementación del *thread* Servidor)
- RefCliente.java (implementación del cliente)
- refranes.txt (Archivo de refranes)
- Entrada.java (implementación de entrada/salida)

El Código de los programas se muestra a continuación

RefServidor.java

```
class RefServidor {
   public static void main(String[] args) {
      new RefServidorThread().start();
   }
}
```

RefServidorThread.java

```
import java.io.*;
import java.net.*;
import java.util.*;

class RefServidorThread extends Thread {
   private DatagramSocket socket = null;
   private BufferedReader qfs = null;
   private boolean moreQuotes = true;

   RefServidorThread() {
      super("RefServidor");
      try {
         socket = new DatagramSocket();
         System.out.println("Servidor de Refranes a la escucha en el puerto : "
+ socket.getLocalPort());
      } catch (java.io.IOException e) {
         System.err.println("Es imposible crear un datagram socket.");
      }
      this.openInputFile();
   }

   public void run() {
      if (socket == null)
         return;
```

```
        while (moreQuotes) {
            try {
                byte[] buf = new byte[256];
                DatagramPacket packet;
                InetAddress address;
                int port;
                String dString = null;

                    // receive request
                packet = new DatagramPacket(buf, 256);
                socket.receive(packet);
                address = packet.getAddress();
                port = packet.getPort();

                    // send response
                if (qfs == null)
                    dString = new Date().toString();
                else
                    dString = getNextQuote();
                buf = dString.getBytes();
                packet = new DatagramPacket(buf, buf.length, address, port);
                socket.send(packet);
            } catch (IOException e) {
                System.err.println("IOException:  " + e);
                moreQuotes = false;
                e.printStackTrace();
            }
        }
          socket.close();
    }

    private void openInputFile() {
        try {
            qfs = new BufferedReader(new InputStreamReader(new
FileInputStream("one-liners.txt")));
        } catch (java.io.FileNotFoundException e) {
            System.err.println("Nos se puedo abrir el fichero de refranes. Se sirve
la hora.");
        }
    }
    private String getNextQuote() {
        String returnValue = null;
        try {
            if ((returnValue = qfs.readLine()) == null) {
                qfs.close();
                moreQuotes = false;
                returnValue = "Se acabaron los refranes. Adios.";
```

```
            }
        } catch (IOException e) {
            returnValue = "Se la lanzado una IOException en el servidor.";
        }
        return returnValue;
    }

}
```

RefCliente.java

```
import java.io.*;
import java.net.*;
import java.util.*;

public class RefCliente {
    public static void main(String[] args) throws IOException {

        if (args.length != 1) {
            System.out.println("Uso: java QuoteClient <hostname>");
            return;
        }

        System.out.print("introduzca el numero del puerto: ");
        int puerto = MyInput.readInt();

            // get a datagram socket
        DatagramSocket socket = new DatagramSocket();

            // send request
        byte[] buf = new byte[256];
        InetAddress address = InetAddress.getByName(args[0]);
        DatagramPacket packet = new DatagramPacket(buf, buf.length, address,
puerto);
        socket.send(packet);

            // get response
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);

              // display response
        String received = new String(packet.getData());
        System.out.println("Refran: " + received);

        socket.close();
    }
}
```

Archivo refranes.txt

Quien no buscó amigos en la alegría, en la desgracia no los pida.
La memoria es como el mal amigo; cuando más falta te hace, te falla.
La probabilidad de hacer mal se encuentra cien veces al día; la de hacer bien una vez al año.
No entres donde no puedas pasar fácilmente la cabeza.
Cuando fuiste martillo no tuviste clemencia, ahora que eres yunque, ten paciencia.
Octubre vinatero, padre del buen cuero.
Mezcladas andan las cosas: junto a las ortigas nacen las rosas.
No hay forastero que venga de mala gente, ni viejo que no haya sido valiente.
Nunca es tarde para bien hacer; haz hoy lo que no hiciste ayer.
Octubre vinatero, padre del buen cuero.
Mezcladas andan las cosas: junto a las ortigas nacen las rosas.
No hay forastero que venga de mala gente, ni viejo que no haya sido valiente.
Ira de hermanos, ira de diablos.
A buen entendedor, pocas palabras bastan
A buena hambre no hay pan duro
A caballo regalado, no le mires el diente
A cada cerdo le llega su San Martín
A camino largo, paso corto
A Dios rogando y con el mazo dando
A enemigo que huye, puente de plata
A falta de pan buenas son tortas
A grandes males, grandes remedios.
A la larga el galgo a la liebre mata.
A la ocasión la pintan calva.
A la tercera va la vencida.
A la vejez, viruelas.
A lo hecho, pecho.
A mal tiempo, buena cara.
A palabras necias, oídos sordos.
A perro flaco todo son pulgas.
A perro viejo no hay tus toses.
A quien Dios no le dio hijos, el diablo le dio sobrinos.
A quien Dios se la diere, San Pedro se la bendiga.
A quien madruga, Dios le ayuda.
A rey muerto, rey puesto.
A río revuelto, ganancia de pescadores.
Abril, aguas mil.
Afortunado en el juego, desgraciado en amores.
Agosto, frío en rostro.
Agua pasada no mueve molinos.
Agua por mayo, pan para todo el año.
Agua por San Juan, quita vino y no da pan
Al burro muerto, cebada al rabo.

Al buen callar llaman Sancho.
Al buen pagador no le duelen prendas.
En cien años todos seremos calvos.
Al freír será al reír
Al pan, pan, y al vino, vino.
Al que al cielo escupe, en la cara le cae.
Alábate, cesto, que venderte quiero.
Algo tendrá el agua cuando la bendicen.
Allá van leyes do quieren reyes
Amigo reconciliado, enemigo doblado
Amor con amor se paga
Ande yo caliente ríase la gente
Antes que te cases, mira lo que haces
Año de nieves, año de bienes
Aramos, dijo la mosca al buey.
Aún no ensillamos y ya cabalgamos
Aunque la mona se vista de seda, mona se queda

Entrada.java

```
import java.io.*;
import java.util.*;

public class Entrada
{
  static private StringTokenizer stok;
  static private BufferedReader br
    = new BufferedReader(new InputStreamReader(System.in), 1);

  public static int readInt()
  {
    int i = 0;
    try
    {
      String str = br.readLine();
      StringTokenizer stok = new StringTokenizer(str);
      i = new Integer(stok.nextToken()).intValue();
    }
    catch (IOException ex)
    {
      System.out.println(ex);
    }
    return i;
  }

  public static double readDouble()
  {
```

```
    double d = 0;
    try
    {
      String str = br.readLine();
      stok = new StringTokenizer(str);
      d = new Double(stok.nextToken()).doubleValue();
    }
    catch (IOException ex)
    {
      System.out.println(ex);
    }
    return d;
  }
}
```

1. Compile y ejecute los programas mostrados
2. Implementar un servidor y un cliente de ``eco'' usando las clases DatagramPacket y DatagramSocket (UDPServidor.java y UDPClient e.java).

El código de los programas para la implementación es:

UDPServidor.java

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

/**
  * Esta clase implementa el Servidor de eco con Datagramas.
  **/
public class UDPServidor{
  public static void main(String args[]){
    try {
      // Crear un socket de Datagramas asociado a un puerto determinado
      // Crear el buffer de bytes para almacenar el mensaje entrante

      // Establecer un bucle infinito para dar entrada a los mensajes
        // Crear un objeto Datagrama para almacenar la solicitud
          // Recibir el mensaje

        // Crear un objeto Datagrama con la respuesta
          // Enviar la respuesta
      // Cerrar el socket
    }
    catch (SocketException e) {
```

```
      System.out.println("Socket: " + e.getMessage());
    }
    catch (IOException e) {
      System.out.println("IO: "+ e.getMessage());
    }
  }
}
```

UDPCliente.java

```
public class UDPCliente{
  public static void main(String args[]){
    try {
      // Crear un socket de Datagramas
      // Crear el mensaje a transmitir (args[0])
      // Crear una dirección de Internet (args[1])
      // Representar el puerto en el que está a la escucha el servidor

      // Crear un objeto DatagramPacket con
      //              el mensaje, su longitud, la dirección de Intenet, y el puerto
      // Enviar el mensaje

      // Crear un objeto DatagramPacket para recibir la respuesta del servidor
      // Recibir el mensaje

      // Mostrar e mensaje por pantalla
      // Cerrar el socket
    }
    catch (SocketException e) {
      System.out.println("Socket: " + e.getMessage());
    }
    catch (IOException e) {
      System.out.println("IO: "+ e.getMessage());
    }
  }
```

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.

- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía *(formato APA).*

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 5.

II.- Nombre de la Práctica.

Programa del lado del servidor con sockets

III.- Competencia(s) a desarrollar:

- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.
- Utilizar sockets en el desarrollo aplicaciones cliente/servidor
- Identificar las características de la interfaz socket.
- Utilizar sockets en el desarrollo de aplicaciones básicas cliente/servidor.
- Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones en una red de computadoras:

V.- Introducción.

Los sockets son básicamente formas en las que podemos interconectar 2 (o más) programas mediante el uso de la internet. En java se utilizan para poder crear conexiones utilizando una IP/hostname y un puerto para establecer la conexión. Para aprender podemos utilizarla para conectar 2 programas por medio de Internet.

¿Cómo funciona?

El modelo más básico de los sockets consta de 2 simples programas, un servidor y un cliente. El programa servidor comienza a "escuchar" en un puerto determinado (nosotros lo especificamos), y posteriormente el programa que la hace de "cliente" debe conocer la ip o nombre de dominio/hostname del servidor y el puerto que está escuchando, al saber esto simplemente solicita establecer una conexión con el servidor. Es aquí cuando el servidor acepta esa conexión y se puede decir que estos programas están "conectados", de este modo pueden intercambiar información.

Notas:

*Ambos programas (servidor y cliente) no necesitan estar programados en Java, es posible programarlos en lenguajes de programación diferentes, o inclusive programar un servidor en java y utilizar un cliente ya existente que pueda conectarse a un puerto especificado.

*El cliente debe de conocer tanto el puerto a utilizar como la IP o dominio del servidor, mientras el servidor solo debe conocer el puerto de conexión

A continuación, les dejo este código de un servidor muy simple que yo hice en Java, lo que hace es escuchar el puerto 5000, cuando un cliente se conecta este envía un mensaje de confirmación al cliente, luego el cliente al enviar su primer

mensaje, el servidor envía un segundo mensaje y se cierra la conexión. También el de un cliente muy muy simple solo para que se vea la diferencia, de cómo se pide la conexión al servidor, el cliente únicamente envía un mensaje de texto pero no recibe información.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

2. Programación Cliente Servidor de Bajo Nivel: sockets y canales

Subtemas:

2.1. Concepto de socket.
2.2. Dominios y Tipos de sockets.
2.3. Creación/ implementación y supresión de sockets.
2.4. Desarrollo del lado del servidor con sockets.
2.5. Desarrollo del lado del cliente con sockets

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.

XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red local de computadoras.
- Apuntes de Java.

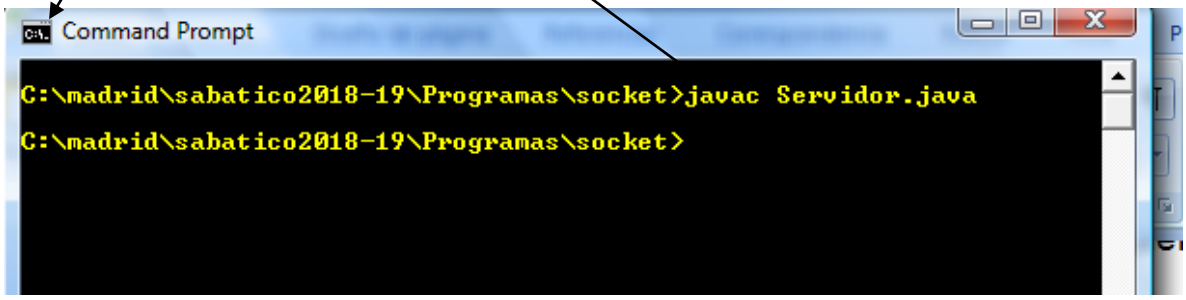
VIII.- Metodología.

En presente práctica se muestra un programa servidor con sockets, describo el código el código, y en la siguiente práctica se hace una prueba en el cual la conexión es exitosa.

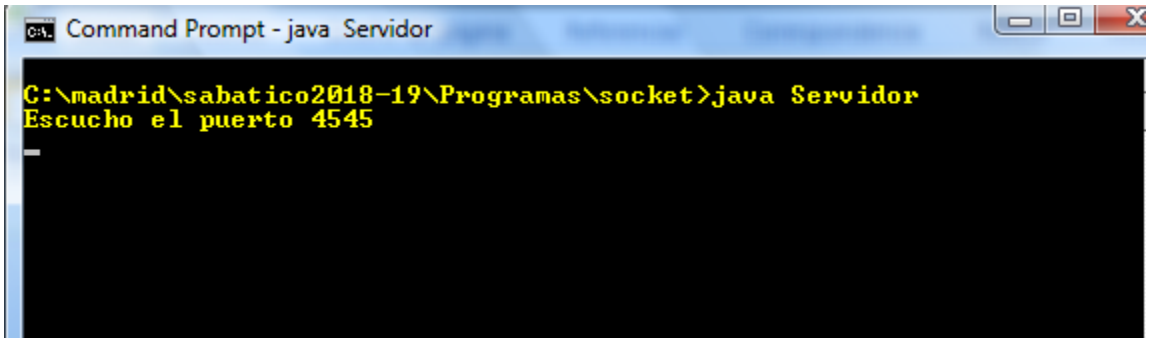
Servidor.java

```
import java.net.* ;
class Servidor {
  static final int PUERTO=4545;
  public Servidor( ) {
    try {
      ServerSocket skServidor = new ServerSocket( PUERTO );
      System.out.println("Escucho el puerto " + PUERTO );
      for ( int numCli = 0; numCli < 5; numCli++ ) {
        Socket skCliente = skServidor.accept(); // Crea objeto
        System.out.println("Sirvo al cliente " + numCli);
        OutputStream aux = skCliente.getOutputStream();
        DataOutputStream flujo= new DataOutputStream( aux );
        flujo.writeUTF( "Hola cliente " + numCli );
        skCliente.close();
      }
      System.out.println("Demasiados clientes por hoy");
    } catch( Exception e ) {
      System.out.println( e.getMessage() );
    }
  }
  public static void main( String[] arg ) {
    new Servidor();
  }
}
```

- ✓ Escriba el código en un editor de texto plano
- ✓ Inicie una sesión en el modo comando (msdos)
- ✓ Compile el programa con ***javac Servidor.java***



- ✓ Ejecute el programa con ***java Servidor***



- ✓ Y ya el Servidor está listo escuchando por el puerto que le indicamos. (4545)

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el

título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía *(formato APA).*

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 6.

II.- Nombre de la Práctica.

Aplicaciones Cliente

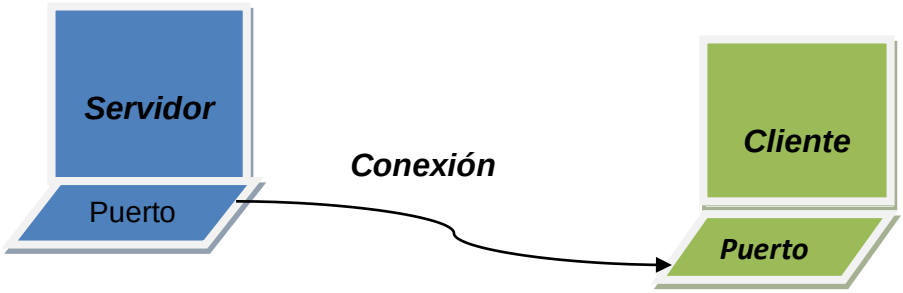
III.- Competencia(s) a desarrollar:

- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.
- Utilizar sockets en el desarrollo aplicaciones cliente/servidor
- Identificar las características de la interfaz socket.
- Utilizar sockets en el desarrollo de aplicaciones básicas cliente/servidor.
- Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones en una red de computadoras:

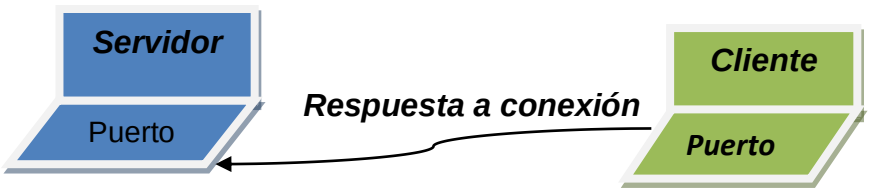
IV.- Introducción.

Sockets

Normalmente, un servidor se ejecuta en una máquina específica y tiene un *socket* asociado a un número de puerto específico. El servidor simplemente espera a la escucha en el *socket* a que un cliente se conecte con una petición. El cliente conoce el *nombre de la máquina* sobre la que está ejecutándose el servidor y el *número de puerto* al que está conectado. Solicitar una conexión consiste en intentar establecer una cita con el servidor en el puerto de la máquina servidora.



Si todo va bien, el servidor acepta la conexión. Pero antes, el servidor crea un nuevo *socket* en un puerto diferente. Es necesario crear un nuevo *socket* (y consecuentemente un número de puerto diferente) de forma que en el *socket* original se continúe a la escucha de las peticiones de nuevos clientes mientras se atiende a las necesidades del cliente conectado. En el cliente, si se acepta la conexión, el *socket* se crea satisfactoriamente y se puede utilizar para comunicarse con el servidor.



Un *socket* es el extremo final de un enlace punto-a-punto que comunica a dos programas ejecutándose en una red.

Los *sockets* siempre están asociados a un número de puerto que es utilizado por TCP para identificar la aplicación a la que está destinada la solicitud y poder redirigírsela.

La clase Socket

La clase Socket del paquete java.net es fácil de usar comparada con la que proporcionan otros lenguajes. Java oculta las complejidades derivadas del establecimiento de la conexión de red y del envío de datos a través de ella. En esencia, el paquete java.net proporciona la misma interfaz de programación que se utiliza cuando se trabaja con archivos.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

2. Programación Cliente Servidor de Bajo Nivel: sockets y canales

Subtemas:

2.1. Concepto de socket.
2.2. Dominios y Tipos de sockets.
2.3. Creación/ implementación y supresión de sockets.
2.4. Desarrollo del lado del servidor con sockets.
2.5. Desarrollo del lado del cliente con sockets

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.

X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red Local
- Apuntes de Java

VIII.- Metodología.

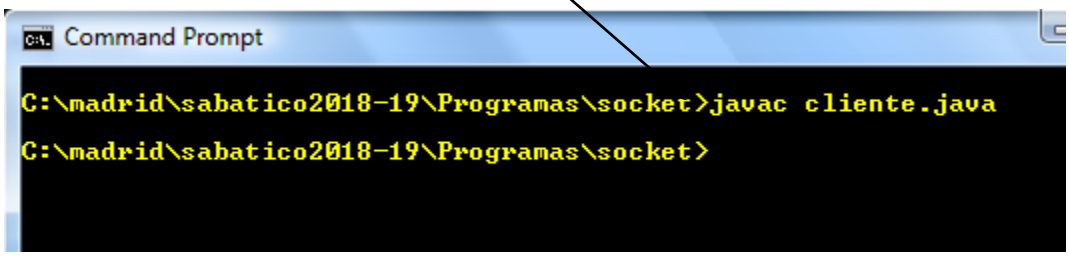
El siguiente código Cliente.java, muestra la implementación de un cliente que accede realiza peticiones de conexión a un servidor. El servidor concreto al que se conecta es al ``localhost'' por convenio, siempre está en el puerto 4545. Lo que ocurre es que el software del servidor está ejecutándose continuamente en la máquina remota, esperando cualquier tráfico de red que ``hable con él'' en el puerto 4545. Cuando el Sistema Operativo de este servidor recupera un paquete de red que contiene una petición para conectar con el puerto 4545, activa el servicio de escucha del servidor y establece la conexión, que permanece activa hasta que es finalizada por alguna de las dos partes.

Cliente.java

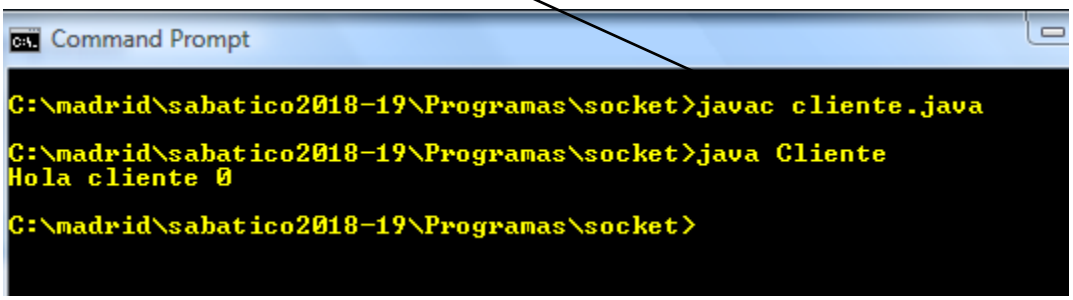
```
import java.io.*;
import java.net.*;
class Cliente {
  static final String HOST = "localhost";
  static final int PUERTO=4545;
  public Cliente( ) {
```

```
    try{
      Socket skCliente = new Socket( HOST , PUERTO );
      InputStream aux = skCliente.getInputStream();
      DataInputStream flujo = new DataInputStream( aux );
      System.out.println( flujo.readUTF() );
      skCliente.close();
    } catch( Exception e ) {
      System.out.println( e.getMessage() );
    }
  }
  public static void main( String[] arg ) {
    new Cliente();
  }

}
```

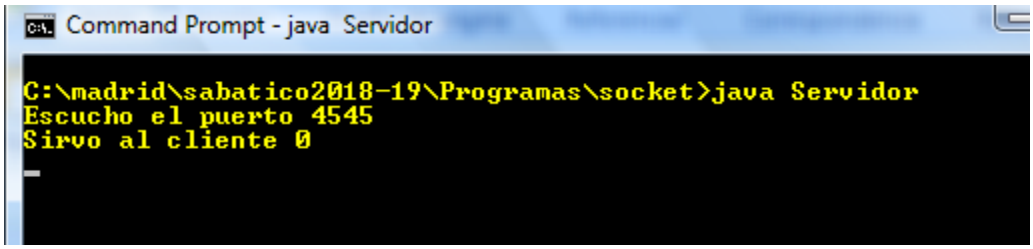
- ✓ Abra otra pantalla de comandos (msdos)
- ✓ Capture el código anterior en un editor de texto plano.
- ✓ Compile el código con ***javac Cliente.java***



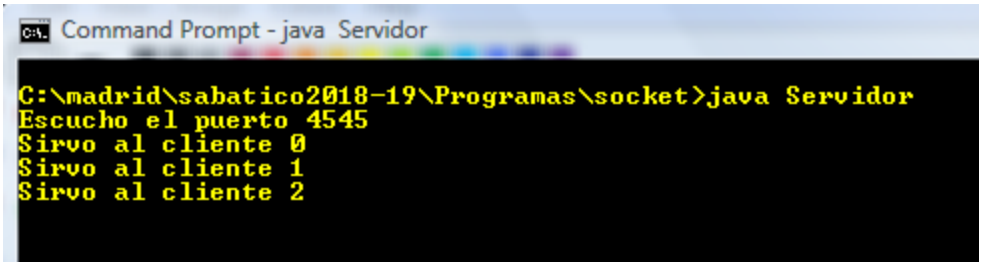
- ✓ El compilador genera el archivo cliente.class
- ✓ Ejecute el programa j***ava Cliente***



- ✓ El servidor ya contesto



- ✓ Se pueden seguir haciendo más conexiones desde diferentes computadoras especificando la dirección ip del servidor y el puerto 4545.



IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía *(formato APA).*

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute , Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 7.

II.- Nombre de la Práctica.

Chat

III.- Competencia(s) a desarrollar:

- Conocer, diseñar y desarrollar aplicaciones atendiendo la arquitectura cliente servidor.
- Utilizar sockets en el desarrollo aplicaciones cliente/servidor
- Identificar las características de la interfaz socket.
- Utilizar sockets en el desarrollo de aplicaciones básicas cliente/servidor.
- Desarrollar aplicaciones utilizando la interfaz socket para la comunicación entre aplicaciones en una red de computadoras:

IV.- Introducción.

Qué es Chat:

El término chat se refiere a un tipo de comunicación digital que se da a través de la red de Internet entre dos o más personas o usuarios. La comunicación por chat se puede llevar a cabo tanto por mensajes de texto, video llamadas o audiochat.

La palabra chat es un término anglosajón que se deriva de chatter, término en inglés que significa "conversación o charla".

Ahora bien, por el gran uso de la palabra chat en la lengua española, incluso se ha generado el verbo chatear que indica la acción de estar siendo partícipe, como usuario, de un chat bien sea público o privado.

Por ejemplo “Esta tarde después de almorzar voy a chatear con mi amiga que vive en Hermosillo”. “Todas las noches antes de dormir chateo con mi mejor amiga”.

El chat, como medio de comunicación, ha sido uno de los grandes avances de las tecnologías de la información y comunicación que ha logrado que millones de personas se comuniquen al instante sin importar las distancias o diferencias de horario.

Los chats son conversaciones que se llevan a cabo al instante gracias al uso de un software conectado a una red de Internet y se diferencian por ser chats públicos, grupos de conversación en los cuales puede participar cualquier persona o, chats privados, que solo pueden participar usuarios autorizados.

El objetivo primordial de los chats es lograr que las personas se comuniquen e intercambien información al instante sin importar donde se localicen, acortando de

esta manera las barreras de la distancia y el tiempo de manera mucho más económica.
Los chats fueron creados como un canal de comunicación e intercambio de información, sobre todo en la actualidad, donde las personas, se conectan constantemente a las redes para estar al tanto acerca de lo que ocurre en el mundo y para adquirir o compartir información.

EJEMPLO CHAT EN JAVA

Hacer un chat para comunicarse entre 2 es sencillo siempre y cuando se entienda bien cómo funcionan los sockets que dicho sea de paso no tienen mucha complicación.

Socket designa a un concepto abstracto por el cual dos programas situados en dos equipos distintos pueden intercambiar cualquier flujo de datos de manera fiable y ordenada, en internet constituye el mecanismo para la entrega de paquetes de datos provenientes de una computadora para que dos programas puedan comunicarse entre si es necesario ciertos requisitos.

Cuando un servidor se ejecuta sobre un equipo tiene un socket que responde a un puerto, este lo único que hace esperar a través de un socket a que un cliente haga una petición.

El cliente conoce el nombre del servidor y el número del puerto en el cual el servidor está conectado, para realizar una petición de conexión el cliente intenta encontrar al servidor.

Para hacer la práctica del chat vamos a hacer 2 aplicaciones independientes, una actuara de servidor y la otra de cliente.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

2. Programación Cliente Servidor de Bajo Nivel: sockets y canales

Subtemas:

- 2.1. Concepto de socket.
- 2.2. Dominios y Tipos de sockets.
- 2.3. Creación/ implementación y supresión de sockets.
- 2.4. Desarrollo del lado del servidor con sockets.
- 2.5. Desarrollo del lado del cliente con sockets

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red Local
- Apuntes de Java

VIII.- Metodología.

Para hacer el chat vamos a hacer 2 aplicaciones independientes, una actuara de servidor y la otra de cliente.

Servidor.

Necesitamos los siguientes atributos:

```
public class Servidor {

    private Socket miServicio;
    private ServerSocket socketServicio;
    private OutputStream outputStream;
    private InputStream inputStream;
    private DataOutputStream salidaDatos;
    private DataInputStream entradaDatos;
    private boolean opcion = true;

    private Scanner scanner;
    private String escribir;
```

- Socket: Para recibir y enviar flujos de datos por la red
- ServerSocket: Para aceptar las conexiones del cliente.
- DataInputStream: Flujo de entrada para procesar los datos enviados por el cliente
- DataOutputStream: Flujo de salida que procesa los datos que se enviarán al cliente.
- Scanner: Para leer datos desde teclado.
- Booleano: Para manipular el hilo.

Método de conexión

Lo primero que tenemos que hacer es poner al ServerSocket a escuchar para poder aceptar peticiones del cliente, para ello vamos a crear un método pasándole como parámetro un numero el cual será el número del puerto.

```
public void conexion(int numeroPuerto) {
        try {
            socketServicio = new ServerSocket(numeroPuerto);
            System.out.println("El servidor se esta escuchando en el puerto: " +
numeroPuerto);
            miServicio = socketServicio.accept();

            //AQUI FALTAN METODOS

        } catch (Exception ex) {
            System.out.println("Error al abrir los sockets");
        }
    }
```

Usamos el constructor Serversocket(numeroPuerto) el cual se le debe pasar como parámetros un numero de puerto donde se pondrá a escuchar recibir peticiones de los clientes y llamamos a su método accept() el cual cuando se conecte un cliente aceptara la petición de dicho cliente, hay que tener en cuenta que estas clases lanzan excepción por lo que hay capturarlo entre un try y un catch, cerramos conexión.

Con el Serversocket a la escucha e el puerto indicado tenemos que abrir los flujos de datos para procesar la información que recibimos y enviamos para ello creamos dos métodos uno que enviara datos y otro que recibirá.

Método enviar datos.

```
public void conexion(int numeroPuerto) {
        try {
            socketServicio = new ServerSocket(numeroPuerto);
            System.out.println("El servidor se está escuchando en el puerto: " +
numeroPuerto);
            miServicio = socketServicio.accept();

            //AQUI FALTAN METODOS

        } catch (Exception ex) {
            System.out.println("Error al abrir los sockets");
        }
    }
```

Se le pasara como parámetros un String el cual usara para recibir cadenas de texto para enviárselos al cliente, instanciamos la clase DataOutputStream el cual le pasaremos como parámetro un OutputStream el cual se usara con método llamado desde el socket misServicio, escribimos en el con el método writeUTF(datos) de la clase DataOutputStream el cual le pasaremos como parámetro el String del método enviarDatos, limpiamos con el método flush() y capturamos su excepción.

Método recibir datos.

```
public void recibirDatos() {
        try {
            inputStream = miServicio.getInputStream();
            entradaDatos = new DataInputStream(inputStream);
            System.out.println(entradaDatos.readUTF());
        } catch (IOException ex) {
            Logger.getLogger(Cliente.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
```

Es un método el cual mediante la clase DataInputStream transmitirá los datos del servidor al cliente abres un flujo recibes los datos y los muestras por pantalla.

Método cerrar todo

```
public void cerrarTodo() {
   try {
      salidaDatos.close();
      entradaDatos.close();
      socketServicio.close();
      miServicio.close();

   } catch(IOException ex) {
      Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null,
ex);
   }
}
```

Este método cerraremos todo lo que hayamos abierto el flujo entrada y salida el SocketServer y el Socket con el método close()

Hilo

Para que esto funcione tendremos que crear en el método conexión un hilo que se encargue de recibir la información que le envié el Cliente usando el método recibirDatos de la clase Servidor e iniciando el hilo que ejecutara cada vez que lleguen datos del cliente.

```
Thread hilo = new Thread(new Runnable() {
@Override
 public void run() {
   while (opcion) {
      System.out.print("SERVIDOR: ");
      recibirDatos();
   }
 }
 });
hilo.start();
```

Scanner

En esta parte del código escribiremos por consola usando la clase Scanner para enviar la información usando el método enviarDatos() todo esto lo encerramos en un while() para poder repetir este proceso hasta que decidamos poner fin a la conversación, en el usaremos un boobleano para controlar su estado.

```
while (opcion){
    scanner = new Scanner(System.in);
    escribir = scanner.nextLine();
    if (!escribir.equals("CLIENTE: fin")) {
        enviarDatos("SERVIDOR: " + escribir);
    } else {
        opcion = false;
    }
}
```

Main

Y para finalizar en el main llamamos a los métodos de nuestra clase para hacer funcionar al servidor.

```
public static void main(String[] args) {
    Servidor serv = new Servidor();
        serv.conexion(5555);
}
```

Programa completo Servidor

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Servidor {

    private Socket miServicio;
    private ServerSocket socketServicio;

    private OutputStream outputStream;
    private InputStream inputStream;

    private DataOutputStream salidaDatos;
    private DataInputStream entradaDatos;

    private boolean opcion = true;
    private Scanner scanner;
    private String escribir;
```

```
   //APERTURA DE SOCKET
   public void conexion(int numeroPuerto) {
      try {
         socketServicio = new ServerSocket(numeroPuerto);
         System.out.println("El servidor se esta escuchando en el puerto: " +
numeroPuerto);
         miServicio = socketServicio.accept();
         Thread hilo = new Thread(new Runnable() {
            @Override
            public void run() {
               while (opcion) {
                  System.out.print("SERVIDOR: ");
                  recibirDatos();

               }
            }
         });
         hilo.start();
         while (opcion) {
            scanner = new Scanner(System.in);
            escribir = scanner.nextLine();
            if (!escribir.equals("CLIENTE: fin")) {
               enviarDatos("SERVIDOR: " + escribir);
            } else {
               opcion = false;
            }
         }
         miServicio.close();
      } catch (Exception ex) {
         System.out.println("Error al abrir los sockets");
      }
   }

   public void enviarDatos(String datos) {
      try {
         outputStream = miServicio.getOutputStream();
         salidaDatos = new DataOutputStream(outputStream);
         salidaDatos.writeUTF(datos);
         salidaDatos.flush();
      } catch (IOException ex) {
         Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null,
ex);
      }

   }
   public void recibirDatos() {
```

```
        try {
            inputStream = miServicio.getInputStream();
            entradaDatos = new DataInputStream(inputStream);
            System.out.println(entradaDatos.readUTF());
        } catch (IOException ex) {
            Logger.getLogger(Cliente.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public static void main(String[] args) {
    Servidor serv = new Servidor();
        serv.conexion(5555);
    }
}
```

Programa Completo Cliente

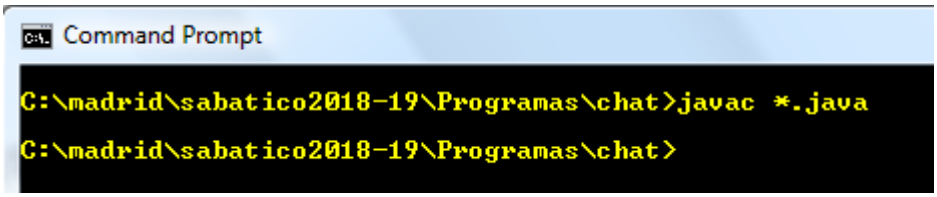
```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Cliente {

    private Socket socketCliente;
    private InputStream inputStream;
    private OutputStream outputStream;
    private DataInputStream entradaDatos;
    private DataOutputStream SalidaDatos;
    private boolean opcion = true;
    private Scanner scanner;
    private String escribir;

    public void conexion(int numeroPuerto, String ipMaquina) {
        try {
            socketCliente = new Socket(ipMaquina, numeroPuerto);
            Thread hilo1 = new Thread(new Runnable() {
                @Override
                public void run() {
                    while (opcion) {
```

```
                escucharDatos(socketCliente);
                System.out.print("CLIENTE: ");
            }
        }
    });
    hilo1.start();
    while (opcion) {
        scanner = new Scanner(System.in);
        escribir = scanner.nextLine();
        if (!escribir.equals("SERVIDOR: fin")) {
            enviarDatos("CLIENTE: " + escribir);
        } else {
            opcion = false;
            cerrarTodo();
        }
    }

} catch (Exception ex) {
    System.out.println("ERROR AL ABRIR LOS SOCKETS CLIENTE " +
ex.getMessage());
}
}
public void escucharDatos(Socket socket) {
    try {
        inputStream = socket.getInputStream();
        entradaDatos = new DataInputStream(inputStream);
        System.out.println(entradaDatos.readUTF());
    } catch (IOException ex) {
        Logger.getLogger(Cliente.class.getName()).log(Level.SEVERE, null,
ex);
    }
}
public void enviarDatos(String datos) {
    try {
        outputStream = socketCliente.getOutputStream();
        SalidaDatos = new DataOutputStream(outputStream);
        SalidaDatos.writeUTF(datos);
        SalidaDatos.flush();
    } catch (IOException ex) {
        Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null,
ex);
    }

}
public void cerrarTodo() {
    try {
        SalidaDatos.close();
```

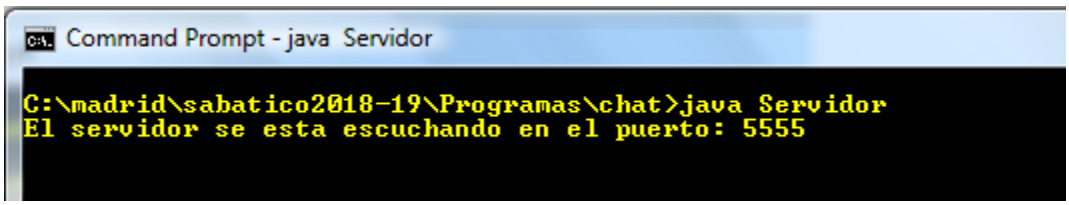
```
            entradaDatos.close();
            socketCliente.close();
        } catch (IOException ex) {
            Logger.getLogger(Cliente.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }


    public static void main(String[] args) {
        Cliente cli = new Cliente();
        cli.conexion(5555, "localhost");
    }
}
```

En esta práctica estamos haciéndolo en la misma máquina por esto estamos conectándonos a localhost. (si lo hacemos desde otra computadora es necesario indicar la dirección IP de la computadora que funge como servidor)

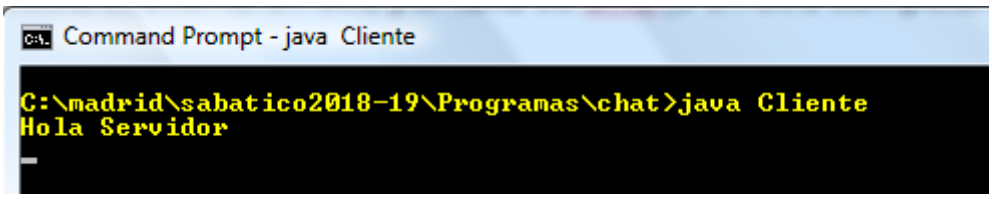
Compile los 2 Programas

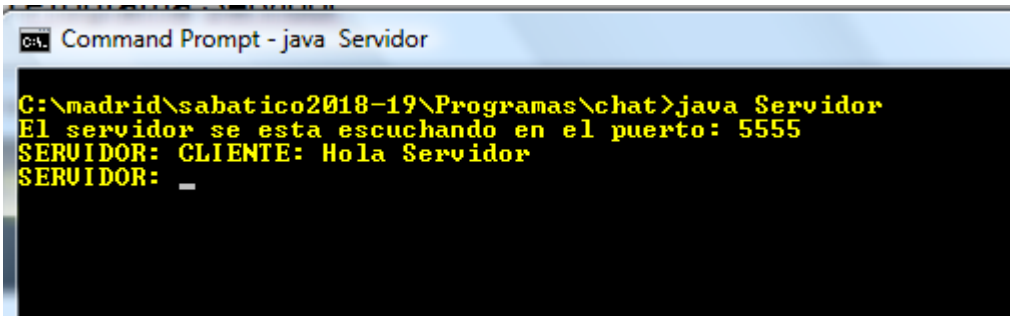


Se Ejecuta en Programa Servidor



Se Ejecuta el Cliente en otra pantalla de cmd (o en otra computadora)





IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (Formato APA).

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 8.

II.- Nombre de la Práctica.

RMI

III.- Competencia(s) a desarrollar:

- Identificar las características, ventajas y desventajas del mecanismo RMI de Java para la intercomunicación de aplicaciones mediante la invocación de métodos remotos.
- Desarrollar aplicaciones empleando el mecanismo RMI.
- Desarrollar programas cliente-servidor utilizando Remote Method Invocation (RMI) como tecnología de base, e incluyendo serialización de objetos, control de políticas de seguridad y generación automática de resguardos
- Seleccionar el modelo de cómputo distribuido pertinente para una aplicación específica.

IV.- Introducción.

Java en Computación Distribuida

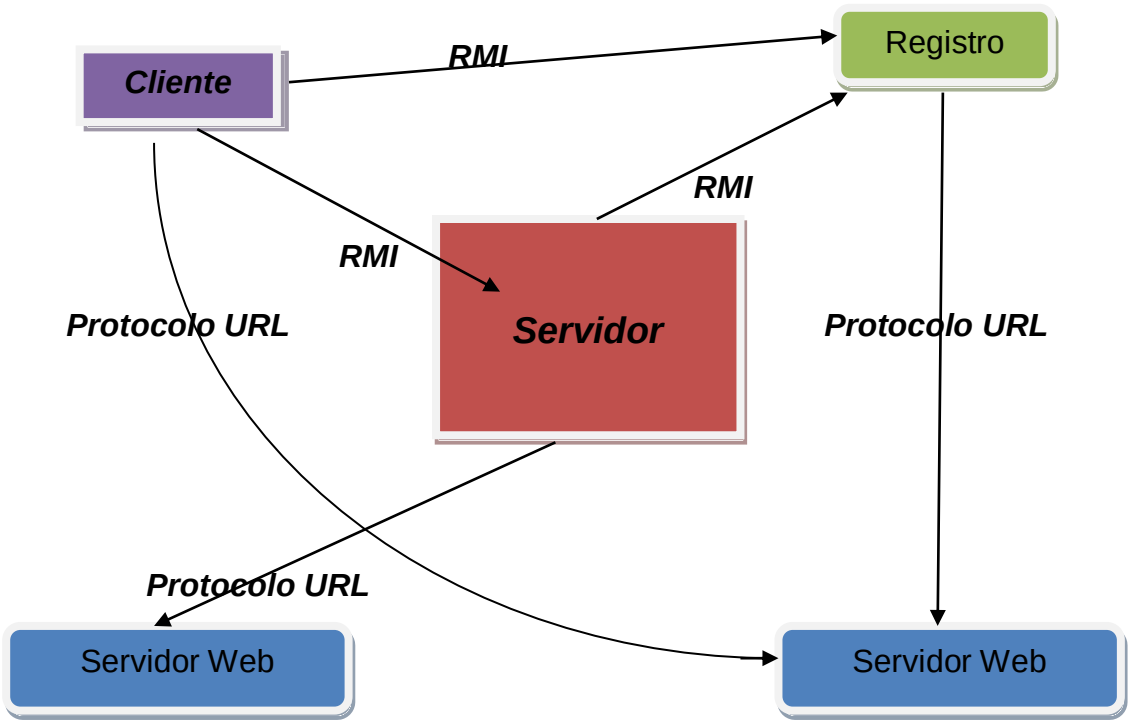
Java es una arquitectura neutral, orientada a objetos, portable y un lenguaje de programación de alto desempeño que proporciona un ambiente de ejecución dinámica, distribuida, robusta, segura y *multi-hilos*. La principal ventaja de Java para computación distribuida radica en la capacidad de descargar el ambiente. En términos de una arquitectura de objeto distribuido totalmente nueva, Java proporciona las siguientes opciones: Java *Remote Method Invocation* (RMI), Java IDL y la empresa *JavaBEan*. La especificación RMI es un API que nos permite crear objetos escritos puramente en lenguaje de programación Java, cuyos métodos se invocan de una Máquina Virtual Java diferente (JVM Java Virtual Machine). La tecnología Java IDL para objetos distribuidos facilitan que los objetos interactúen a pesar de estar escritos en lenguaje de programación Java u otro lenguaje tal como C, C++, COBOL, entre otros.

Java RMI

Básicamente RMI proporciona la capacidad para llamadas a métodos sobre objetos remotos, los cuales convierten al componente de transporte del objeto en arquitectura de objeto distribuido. También proporciona mecanismos para el registro y persistencia del objeto. Ofrece servicios distribuidos tales como Java IDL que proporciona una forma de conectar, transparentemente, a los clientes Java a los servidores de red utilizando la industria estándar: Lenguaje de Definición de Interfaces *(IDL Interface Definition Language)*. Las aplicaciones RMI están, a menudo, compuestas de dos programas separados: un servidor y un cliente. Una aplicación común del servidor crea algunos objetos remotos, realiza referencias para accederlos y se encuentra en espera de que los clientes invoquen los

métodos sobre éstos objetos remotos. Una aplicación del cliente tiene una referencia remota a uno o más objetos remotos en el servidor y entonces invoca al método sobre ellos. RMI proporciona los mecanismos a través de los cuales el servidor y el cliente se comunican e intercambian información. Las aplicaciones utilizan uno o dos mecanismos para obtener referencias a objetos remotos. Una aplicación registra sus objetos remotos con la facilidad de denominación del RMI, o bien la aplicación pasa y regresa la referencia a los objetos remotos como parte de su operación normal. Los detalles de comunicación entre objetos remotos están a cargo del RMI; para el programador, la comunicación remota se asemeja a la invocación del método Java. Dado que RMI permite que un solicitante pase objetos a objetos remotos, RMI proporciona los mecanismos necesarios para cargar un código de objeto, así como también de transmitir sus datos. RMI soporta su propio protocolo de transporte denominado Protocolo de Mensajes Remotos Java *(JRMP Java Remote Messaging Protocol)* para definir el conjunto de formatos de mensajes que permiten que los datos pasen a través de una red de computadoras a otra.

La siguiente ilustración representa una aplicación distribuida RMI que utiliza el registro para obtener una referencia a un objeto remoto. El servidor llama al registro para asociar (o ligar) un nombre con un objeto remoto. El cliente busca el objeto remoto por su nombre en el registro del servidor y entonces invoca un método sobre él. La ilustración también muestra que el sistema RMI utiliza un servidor Web para cargar los códigos en bytes de las clases, del servidor al cliente y del cliente al servidor, para los objetos cuando se les necesita.



RMI pasa objetos por su tipo verdadero permitiendo que nuevos tipos sean introducidos en una máquina virtual remota por consiguiente extendiendo el comportamiento de una aplicación de manera dinámica. RMI trata un objeto remoto de manera diferente de un objeto local cuando el objeto se le pasa de una máquina virtual a otra. En lugar de hacer una copia de la implementación del objeto en la máquina virtual receptora. El solicitante invoca un método en el ***stub*** local que tiene como responsabilidad cargar la llamada al método en el objeto remoto. Un ***stub*** para objetos remotos implementa el mismo conjunto de interfaces remotas que el mismo objeto remoto implementa. Esto permite que un ***stub*** se acople a cualquiera de las interfaces que el objeto remoto implemente. Sin embargo, esto también significa que solamente aquellos métodos, definidos en una interface remota, estén disponibles para su solicitud en la máquina virtual receptora.

¿RMI o IDL?

- Java RMI es una solución 100% Java para objetos remotos que proporciona todas las ventajas de las capacidades de Java "una vez escrito, ejecutarlo en cualquier parte". Los servidores y clientes desarrollados con Java RMI se muestran en cualquier lugar en la red sobre cualquier plataforma que soporta el ambiente de ejecución de Java. Java IDL, en contraste, está basado en una industria estándar para solicitar, de manera remota, a objetos escritos en cualquier lenguaje de programación. Como resultado, Java IDL proporciona un medio para conectar aplicaciones "transferidas" que aún cubren las necesidades de comercio pero que fueron escritos en otros lenguajes.
- Actualmente Java RMI y Java IDL emplean protocolos diferentes para la comunicación entre objetos sobre diferentes plataformas. Java IDL utiliza el protocolo estándar de CORBA IIOP ***(Internet Inter-Orb Protocol).*** Al igual que IDL, IIOP facilita la residencia de objetos en diversas plataformas y escritos en diversos lenguajes para interactuar de manera estándar. Actualmente Java RMI utiliza el protocolo JRMP ***(Java Remote Messaging Protocol),*** un protocolo desarrollado específicamente para los objetos remotos de Java.
- En Java IDL, un cliente interactúa con un objeto remoto por referencia. Esto es, el cliente nunca tiene una copia actual del objeto servidor en su propio ambiente de ejecución. En su lugar, el cliente utiliza ***stubs*** en la ejecución local para manipular el objeto servidor residiendo sobre la plataforma remota. En contraste, RMI facilita que un cliente interactúe con un objeto remoto por referencia, o por su valor descargándolo y manipulándolo en el ambiente de ejecución local. Esto se debe a que todos los objetos en RMI son objetos Java. RMI utiliza las capacidades de ***serialización*** del objeto, que proporciona el lenguaje Java, para transportar los objetos desde el servidor al cliente. Java IDL, dado que interactúa con objetos escritos en cualquier lenguaje, no toma ventaja de "una vez escrito, ejecutarlo en cualquier parte", característica del lenguaje de programación Java.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

3. - RMI (REMOTE METHOD INVOCATION)

Subtemas:

3.1. Características y Estructura de RMI
3.2. El API Java RMI.
3.3. Jerarquía de objetos RMI.
3.4. El Sistema de Nombrado Registry.
3.5. Desarrollo de Aplicaciones Distribuidas. 3.6. Paso de parámetros a través de la red.
3.7. Callbacks (Resguardos).

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario

pida autorización y se haga responsable del equipo.

XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.

XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red Local
- Apuntes de Java

VIII.- Metodología.

***Ejemplo práctico de RMI*. (Java Remote Method Invocation)**
CALCULADORA.

Es hora de construir un sistema que trabaje en RMI y comprender su funcionamiento. En esta práctica, construiremos un simple servicio de calculadora remota y la utilizaremos con un programa cliente.

Un sistema que trabaja en RMI está compuesto de varias partes:

1. Definición de Interfaz para los servicios remotos.
2. Implementación de los servicios remotos.
3. Los archivos resguardo ***(stub).***
4. Host server.
5. Cliente.
6. Instrucciones para compilar y ejecutar el ejemplo.

Pasos para construir el sistema:

1. Escribir y compilar el código java para la interfaz.
2. Escribir y compilar el código java para la implementación de las clases.
3. Compilar los programas Generados.
4. Generar los archivos Stub de la implementación de las clases.
5. Escribir el código java para un servicio de programa host remoto.
6. Desarrollar el código java para un programa cliente RMI.
7. Instalar y correr un sistema RMI.

Definición de la interfaz:

El primer paso al crear y compilar el código para el servicio de interfaz. La interfaz "Calculadora" define todas las características ofrecidas por el servicio:

Calculadora.java

```
public interface Calculadora
  extends java.rmi.Remote {

    public long Suma(long num1, long num2)
                throws java.rmi.RemoteException;

    public long Resta(long num1, long num2)
            throws java.rmi.RemoteException;

    public long Multiplicacion(long num1, long num2)
            throws java.rmi.RemoteException;

    public long Division(long num1, long num2)
            throws java.rmi.RemoteException;
}
```

Implementación de la clase.

El código para la implementación del servicio remoto está en la clase "CalculadoraImpl":

CalculadoraImpl.java

```
public class CalculadoraImpl
  extends java.rmi.server.UnicastRemoteObject implements Calculadora {

 //Las implementaciones deben tener un constructor explícito
 //para declarar la excepción RemoteException

    public CalculadoraImpl()  throws java.rmi.RemoteException {
        super();
    }

    public long Suma(long num1, long num2) throws
java.rmi.RemoteException {
        return nu1 + num2;
    }

    public long Resta(long num1, long num2) throws
java.rmi.RemoteException {
```

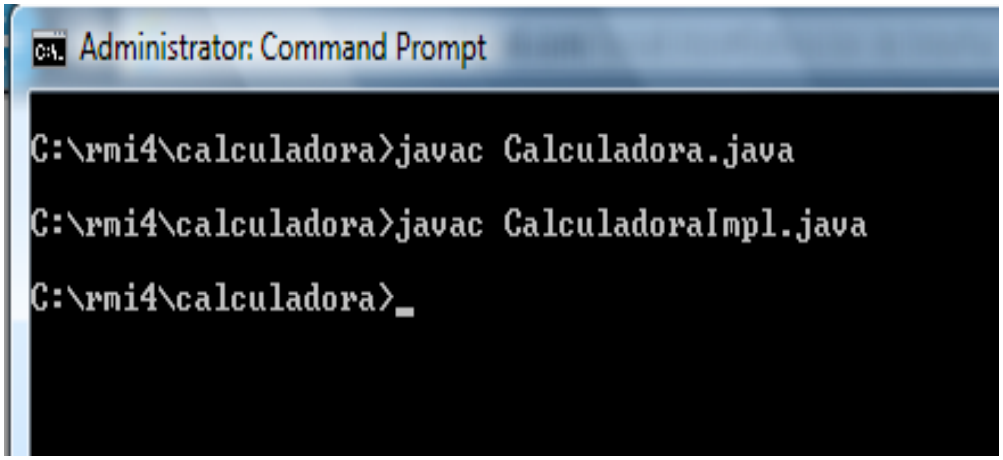
```
            return num1 - num2;
        }

        public long Multiplicacion(long num1, long num2) throws
java.rmi.RemoteException {
            return num1 * num2;
        }

        public long Division(long num1, long num2) throws
java.rmi.RemoteException {
            return num1/ num2;
        }
}
```

Cada objeto contiene las variables de instancia de su clase. Lo que no es tan obvio es que cada objeto también tiene todas las variables de instancia de todas las clases (clase súper padre, abuelo de clase, etc.) Estas variables de clase súper deben inicializar antes de instancias de variables de la clase.

Compilar los programas.

A continuación, Compilamos los archivos generados hasta este momento.

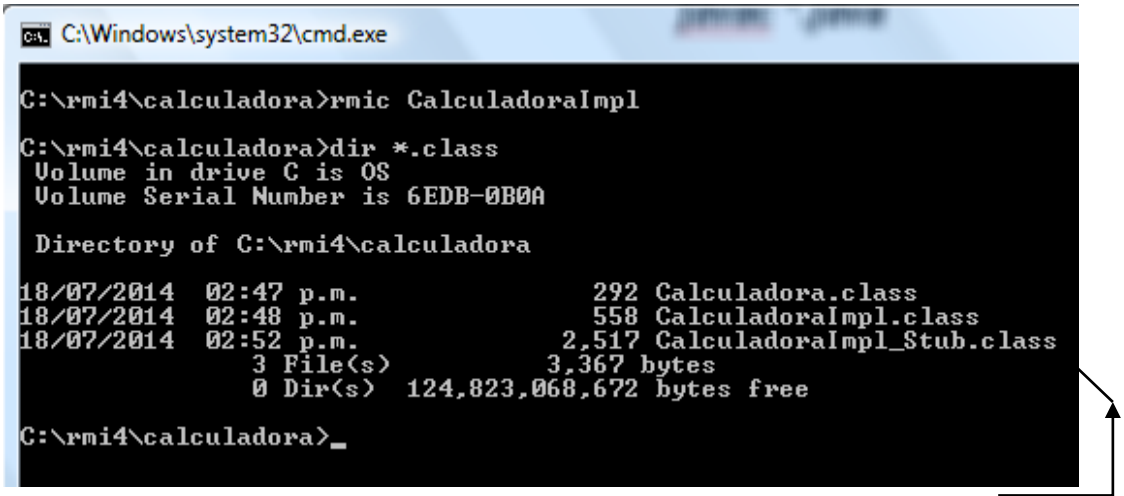
javac *.java



Creación de Archivos stub.

El siguiente paso es utilizar el compilador RMI "rmic", para generar los archivos stub. El compilador corre en el servicio remoto del archivo clase implementación.

c:>rmic4\calculadora>rmic CalculadoraImpl



Realiza esto en el directorio. Después de correr ***rmic*** debemos encontrar el archivo ***Calculadora_Stub.class***.

Archivo aquí

Servidor Anfitrión.

Los servicios remotos RMI deben ser hospedados en un servidor de procesos.

La clase ***"CalculadoraServidor***" es un servidor muy simple:

CalculadoraServidor.java

```
import java.rmi.Naming;

        public class CalculadoraServidor {

          public CalculadoraServidor() {
            try {
              Calculadora c = new CalculadoraImpl();
              Naming.rebind("rmi://localhost:1099/CalculadoraServicio", c);
            } catch (Exception e) {
              System.out.println("Trouble: " + e);
            }
          }
          public static void main(String args[]) {
            new CalculadoraServidor();
          }
}
```

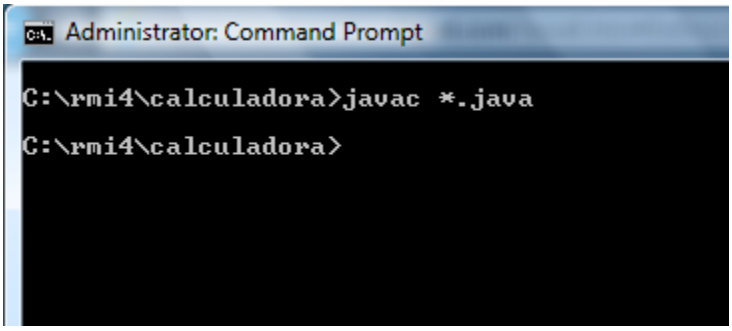
Cliente:

El código fuente para el cliente es el siguiente:

CalculadoraCliente.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;

public class CalculadoraCliente {

  public static void main(String[] args) {
      try {
          Calculadora c = (Calculadora)
              Naming.lookup("rmi://127.0.0.1:1099/CalculadoraServicio");
            System.out.println( c.Suma(4, 3) );
            System.out.println( c.Resta(4, 5) );
            System.out.println( c.Multiplicacion(3, 6) );
            System.out.println( c.Division(9, 3) );
      }
      catch (MalformedURLException murle) {
              System.out.println();
            System.out.println("MalformedURLException");
              System.out.println(murle);
      }
      catch (RemoteException re) {
            System.out.println();
            System.out.println("RemoteException");
              System.out.println(re);
      }
      catch (NotBoundException nbe) {
            System.out.println();
            System.out.println("NotBoundException");
              System.out.println(nbe);
      }
      catch (java.lang.ArithmeticException ae) {
            System.out.println();
            System.out.println("java.lang.ArithmeticException");
              System.out.println(ae);
      }
    }
}
```

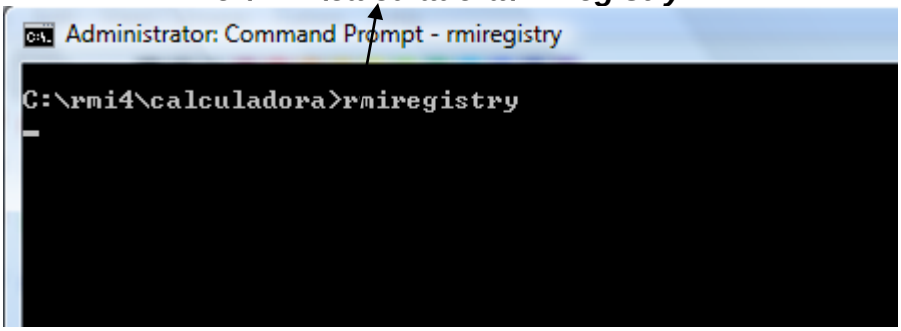
Compilar y Ejecutar el programa.



Ahora estamos listos para correr el sistema. Nosotros necesitamos tres consolas, una para el ***servidor***, una para el ***cliente*** y otra para el ***RMIRegistry.***

Comienza con el ***Registry***. Debemos estar en el directorio que contiene las clases que escribimos. Aquí introducimos lo siguiente:

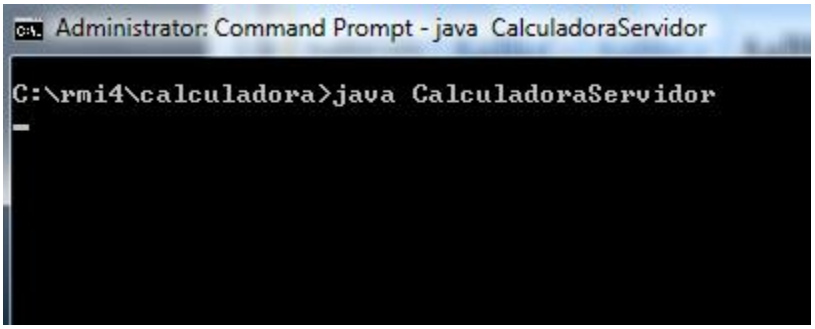
c:\rmi4\calculadora\rmiregistry



El registry ya está corriendo y podremos pasar a la siguiente consola.

En la segunda consola comenzamos el servidor "**CalculadorServidor"** e introducimos lo siguiente:

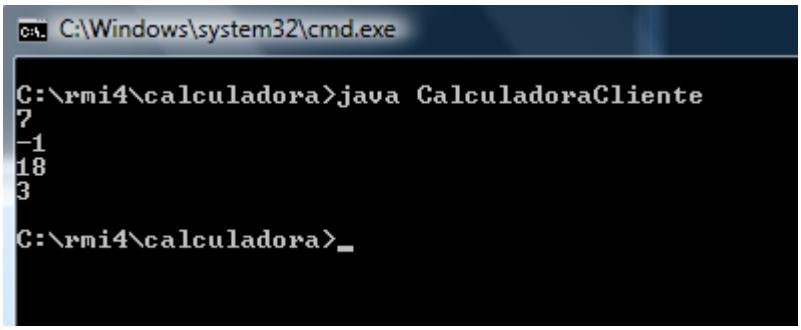
C:\rmi4\calculadora>java CalculadoraServidor



Comenzará cargando la implementación dentro de la memoria y espera por una conexión cliente.

En la última consola, comenzamos el cliente:

C:\rmi4\calculadora>java CalculadoraCliente



Vemos la salida en la pantalla

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

Practica.

Realice una práctica en equipo (3 participantes) que ejecute las operaciones de los valores capturadas por pantalla a un servidor desde 2 clientes (equipos diferentes).

a) Que se ejecuta en el lado del servidor.
b) Que se ejecuta en el lado del cliente.
c) Como se comunicaría (*Naming.lookup)*

Subir los resultados a la plataforma.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el

curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (Formato APA).

1. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
2. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
3. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
4. Wong, Paul. Java. Ed. International Thomsom Editores.
5. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
6. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
7. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 9.

II.- Nombre de la Práctica.

Creación de Servidores COM.

III.- Competencia(s) a desarrollar:

- Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model)
- Seleccionar el modelo de cómputo distribuido pertinente para una aplicación específica.
- Identificar las características, ventajas y desventajas del Modelo de Objetos de Componentes Distribuidos de Microsoft.
- Desarrollar aplicaciones bajo DCOM.

IV.- Introducción.

En las primeras épocas de la computación las computadoras operaban independientemente una de otra sin tener comunicación entre ellas. Las aplicaciones de software eran comúnmente desarrolladas para un propósito específico. Compartir los datos entre sistemas era mínimo y se hacías de una manera muy fácil, se transportaban los medios de almacenamiento (tarjetas, cintas, discos, etc.) De un lado a otro. El próximo paso fue conectar las computadoras a través de una red usando protocolos propietarios, luego los protocolos fueron estandarizados. Luego llego la era de los sistemas abiertos y la integración de sistemas por los cuales un cliente podía elegir varios componentes de hardware de diferentes vendedores e integrarlos para crear una configuración necesaria con costos razonables.

Así siguió el paso de los años. Nuevas técnicas de desarrollo de software se fueron sucediendo una tras otra, desde la programación estructurada y modular hasta la programación orientada a objetos, siempre buscando reducir costos y aumentar la capacidad de reutilización. Si bien la programación orientada a objetos fomentó el reuso y permitió reducir costos, no se ha logrado aún el objetivo último: comprar componentes de varios proveedores e integrarlos para formar aplicaciones que a su vez se integren para formar sistemas completos.
Para lograr la integración total de componentes realizados por terceras partes es necesario llegar a un acuerdo común en el que se establezcan los mecanismos necesarios para que esa integración se haga efectiva. Será necesario especificar de manera independiente al lenguaje de programación en el que se desarrolló el componente cuáles son sus puntos de acceso (funciones), luego será necesario establecer los mecanismos de comunicación entre componentes, que podrían estar ejecutándose en una máquina remota.
En este sentido, y buscando satisfacer esa necesidad de mecanismos estándar e interfaces abiertas, son tres los esfuerzos que más han sobresalido. Por un lado,

Microsoft ha introducido en el mercado sus tecnologías COM, DCOM y COM+. Otro participante es Sun Microsystems, que ha presentado Java Beans. El tercero es el Object Management Group, un consorcio integrado por varias industrias importantes, que ha desarrollado CORBA (Common Request Broker Architecture).

COM / DCOM

Microsoft Distributed COM (DCOM) extiende COM (Component Object Model) para soportar comunicación entre objetos en ordenadores distintos, en una LAN, WAN, o incluso en Internet. Con DCOM una aplicación puede ser distribuida en lugares que dan más sentido al cliente y a la aplicación.

Como DCOM es una evolución lógica de COM, se pueden utilizar los componentes creados en aplicaciones basadas en COM, y trasladarlas a entornos distribuidos. DCOM maneja detales muy bajos de protocolos de red, por lo que uno se puede centrar en la realidad de los negocios: proporcionar soluciones a clientes.

Actualmente DCOM viene con los sistemas operativos Windows 2000, NT, 98 y también está disponible una versión para Windows 95 en la página de Microsoft. También hay una implementación de DCOM para Apple Macintosh y se está trabajando en implementaciones para plataformas UNIX como Solaris.

La arquitectura DCOM

DCOM es una extensión de COM, y éste define como los componentes y sus clientes interactúan entre sí. Esta interacción es definida de tal manera que el cliente y el componente pueden conectar sin la necesidad de un sistema intermedio. El cliente llama a los métodos del componente sin tener que preocuparse de niveles más complejos.

En los actuales sistemas operativos, los procesos están separados unos de otros. Un cliente que necesita comunicarse con un componente en otro proceso no puede llamarlo directamente, y tendrá que utilizar alguna forma de comunicación entre procesos que proporcione el sistema operativo. COM proporciona este tipo de comunicación de una forma transparente: intercepta las llamadas del cliente y las reenvía al componente que está en otro proceso.

Cuando el cliente y el componente residen en distintas máquinas, DCOM simplemente reemplaza la comunicación entre procesos locales por un protocolo de red. Ni el cliente ni el componente se enteran de que la unión que los conecta es ahora un poco más grande.

Las librerías de COM proporcionan servicios orientados a objetos a los clientes y componentes, y utilizan RPC y un proveedor de seguridad para generar paquetes de red estándar que entienda el protocolo estándar de DCOM.

Los Componentes y su reutilización

Muchas aplicaciones distribuidas no están desarrolladas

Al existir infraestructuras de hardware, software, componentes, al igual que herramientas, se necesita poder integrarlas y nivelarlas para reducir el desarrollo y el tiempo de trabajo y coste. DCOM toma ventaja de forma directa y transparente de los componentes COM y herramientas ya existentes. Un gran mercado de todos los componentes disponibles haría posible reducir el tiempo de desarrollo integrando soluciones estandarizadas en las aplicaciones de usuario. Muchos desarrolladores están familiarizados con COM y pueden aplicar fácilmente sus conocimientos a las aplicaciones distribuidas basadas en DCOM.

Cualquier componente que sea desarrollado como una parte de una aplicación distribuida es un candidato para ser reutilizado. Organizando los procesos de desarrollo alrededor del paradigma de los componentes permite continuar aumentando el nivel de funcionalidad en las nuevas aplicaciones y reducir el tiempo de desarrollo.

Diseñando para COM y DCOM se asegura que los componentes creados serán útiles ahora y en el futuro.

Independencia de la localización

Cuando se comienza a implementar una aplicación distribuida en una red reak, aparecen distintos conflictos en el diseño:

- Los componentes que interactúan más a menudo deberían estar localizados más cerca.
- Algunos componentes solo pueden ser ejecutados en máquinas específicas o lugares específicos.
- Los componentes más pequeños aumentan la flexibilidad, pero aumentan el tráfico de red.
- Los componentes grandes reducen el tráfico de red, pero también reducen la flexibilidad.

Con DCOM, estos temas críticos de diseño pueden ser tratados se forma bastante sencilla, ya que estos detalles no se especifican en el código fuente. DCOM olvida completamente la localización de los componentes, ya esté en el mismo proceso que el cliente o en una máquina en cualquier lugar del mundo. En cualquier caso, la forma en la que el cliente se conecta a un componente y llama a los métodos de éste es idéntica. No es solo que DCOM no necesite cambios en el código fuente, sino que además no necesita que el programa sea recompilado. Una simple reconfiguración cambia la forma en la que los componentes se conectan entre sí.

La independencia de localización en DCOM simplifica enormemente la tarea de los componentes de aplicaciones distribuidas para alcanzar un nivel de funcionamiento óptimo. Supongamos, por ejemplo, que cierto componente debe ser localizado en una máquina específica en un lugar determinado. Si la aplicación tiene numerosos componentes pequeños, se puede reducir la carga de la red

situándolos en la misma LAN, en la misma máquina, o incluso en el mismo proceso. Si la aplicación está compuesta por un pequeño número de grandes componentes, la carga de red es menor y no es un problema, por tanto se pueden poner en las máquinas más rápidas disponibles independientemente de donde estén situadas.

Con la independencia de localización de DCOM, la aplicación puede combinar componentes relacionados en máquinas "cercanas" entre si, en una sola máquina o incluso en el mismo proceso. Incluso si un gran número de pequeños componentes implementan la funcionalidad de un gran módulo lógico, podrán interactuar eficientemente entre ellos.

Independencia del lenguaje de programación

Una cuestión importante durante el diseño e implementación de una aplicación distribuida es la elección del lenguaje o herramienta de programación. La elección es generalmente un término medio entre el coste de desarrollo, la experiencia disponible y la funcionalidad. Como una extensión de COM, DCOM es completamente independiente del lenguaje. Virtualmente cualquier lenguaje puede ser utilizado para crear componentes COM, y estos componentes puede ser utilizado por muchos más lenguajes y herramientas. Java, Microsoft Visual C++, Microsoft Visual Basic, Delphi, PowerBuilder, y Micro Focus COBOL interactuan perfectamente con DCOM.

Con la independencia de lenguaje de DCOM, los desarrolladores de aplicaciones pueden elegir las herramientas y lenguajes con los que estén más familiarizados. La independencia del lenguaje permite crear componentes en lenguajes de nivel superior como Microsoft Visual Basic, y después re implementarlos en distintos lenguajes como C++ o Java, que permiten tomar ventaja de características avanzadas como multihilo.

Independencia del protocolo

Muchas aplicaciones distribuidas tienen que ser integradas en la infraestructura de una red existente. Necesitar un protocolo específico de red, obligará a mejorar todos los clientes, lo que es inaceptable en muchas situaciones. Los desarrolladores de aplicaciones tienen que tener cuidado de mantener la aplicación lo más independiente posible de la infraestructura de la red.

DCOM proporciona esta transparencia: DCOM puede utilizar cualquier protocolo de transporte, como TCP/IP, UDP, IPX/SPX y NetBIOS. DCOM proporciona un marco de seguridad a todos estos protocolos.

Los desarrolladores pueden simplemente utilizar las características proporcionadas por DCOM y asegurar que sus aplicaciones son completamente independientes del protocolo.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

4. - COM/DCOM (Component Object Model/ Distributed COM)

Subtemas:

4.1. Creación de Servidores COM.
4.2. Creación de un cliente COM.
4.3. Automatización.
4.4. ATL (Active Template Library).
4.5. DCOM (Distributed COM).

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.

XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red de local.
- Software de Visual Basic.
- Manual de Visual Basic.

VIII.- Metodología.

Este practica demuestra cómo crear, empaquetar e implementar una aplicación cliente-servidor del Modelo distribuido de objetos componentes (DCOM) con Visual Basic. Para crear una aplicación cliente-servidor DCOM, se necesita Visual Basic Enterprise Edition. Es necesario estar familiarizado con la creación de aplicaciones cliente-servidor que se ejecutan en el mismo equipo.

No tiene que cambiar el código para permitir que una aplicación cliente cree instancias de un servidor remoto con DCOM. La diferencia es la manera en la que se empaqueta e implementa el cliente.

Los pasos siguientes demuestran cómo distribuir y configurar una aplicación cliente-servidor simple. Denomine al servidor DCOMServer. Cree una carpeta para el servidor que para esta práctica denominaremos c:\DCOM\Servidor.

Crear el servido

- Inicie un proyecto nuevo de Visual Basic. En el cuadro de diálogo Nuevo proyecto, seleccione EXE ActiveX y, a continuación, haga clic en Abrir. Se creará Class1 de manera predeterminada.
- Agregue el siguiente código al módulo de Class1:

```
Public Function ServerTime() As String
    ServerTime = Time
End Function
```

- En el menú Proyecto, haga clic en la opción Propiedades del proyecto y, a continuación, seleccione la ficha General.
- En el campo Nombre de proyecto, escriba DCOMServer.
- En el campo Descripción del proyecto, escriba DCOMPractica - Servidor. Active la opción Ejecución desatendida.

  NOTA: esta opción siempre se debería activar en los servidores que no

tengan ninguna interfaz de usuario para garantizar que no se muestra ningún cuadro de diálogo de ningún tipo mientras el servidor se está ejecutando. Si existe algún tipo de interacción con el usuario mientras el servidor se está ejecutando bajo una identidad que no es la de Usuario interactivo, puede parecer que el servidor deja de responder.

- Seleccione la ficha Componente y active la opción Archivos en servidor remoto.

  NOTA: al activar esta opción, hace que el compilador de Visual Basic genere los archivos VBR y TLB que se necesitan para empaquetar las aplicaciones cliente que utilizan este servidor. Estos archivos contienen entradas del Registro que tienen que incluirse en el equipo cliente.

- Cierre el cuadro de diálogo Propiedades del proyecto.
- En el menú Archivo, seleccione Guardar como y, a continuación, guarde este proyecto en la carpeta c:\DCOM\Servidor.
- En el menú Archivo, seleccione Generar DCOMServer y compile el servidor.
- En el menú Proyecto, seleccione la opción Propiedades del proyecto y, a continuación, seleccione la ficha Componente.
- En la ficha Componente, seleccione Compatibilidad de la versión, seleccione la opción Compatibilidad binaria y, a continuación, haga que el proyecto tenga compatibilidad binaria con el archivo ejecutable del servidor recién creado (DCOMServer.exe). Al seleccionar esta opción, está garantizando que todos los GUID se mantienen igual si vuelve a compilar el servidor.

Empaquetar el servidor

Utilice el Asistente de empaquetado e implementación para empaquetar como de costumbre el servidor para la distribución. Un cliente remoto crea una instancia del servidor con DCOM. Al crear el paquete para el servidor, obtiene un cuadro de diálogo en el que se pregunta si este servidor se utilizará como servidor remoto de automatización y si desea incluir los archivos de compatibilidad para este propósito. Basta con que haga clic en el botón No, porque DCOM no es un modelo de automatización remota. La automatización remota es una tecnología anterior a la que reemplazó DCOM.

Instalar el servidor

Instale el servidor en el equipo en el que desearía ejecutarlo, utilizando el paquete de la distribución que creó anteriormente. Si desea utilizar el equipo de desarrollo para ejecutar el servidor, no necesita instalarlo porque Visual Basic realiza el registro en su lugar cuando compila el servidor.

Establecer la seguridad del servidor

Si instaló el servidor en un equipo Microsoft Windows NT o Microsoft Windows 2000, tiene que configurar su seguridad. Para ello, use Dcomcnfg según se muestra en los pasos siguientes, en los que se da por supuesto que los equipos cliente y servidor forman parte de un dominio, y el usuario que inició sesión en el equipo cliente lo hizo como usuario del dominio. La configuración sugerida es simplemente una posible. Es muy genérica y proporciona un acceso extenso al servidor. Recuerde que simplemente se trata de un ejemplo. Al implementar aplicaciones reales, si la seguridad es importante para el entorno, debería seleccionar opciones más restrictivas. Además, si el equipo que usa para probar este servidor de ejemplo se utiliza para ejecutar otros servidores, tome nota de la configuración actual antes de realizar los cambios siguientes, y vuelva a la original en cuanto termine las pruebas.

- En el equipo servidor, haga clic en el botón Iniciar y, a continuación, seleccione Ejecutar. En el cuadro de diálogo Ejecutar, escriba Dcomcnfg y, después, haga clic en Aceptar. Debe tener derechos de administrador para poder ejecutar Dcomcnfg.
- Seleccione la ficha Propiedades predeterminadas y compruebe que la opción Habilitar COM distribuido en este equipo está activada.
- Establezca el nivel de autenticación predeterminado en Conectary el nivel de suplantación predeterminado en Identificar.
- Seleccione la ficha Seguridad predeterminada.
- Haga clic en el botón Editar valores predeterminados en el panel Permisos de acceso predeterminados.
- Compruebe que Todos y Sistema están incluidos en la lista con derechos Permitir acceso. Si no lo están, puede utilizar el botón Agregar para agregarlos a la lista. Haga clic en Aceptar cuando la lista esté completa.
- Haga clic en el botón Editar valores predeterminados en el panel Permisos de inicio predeterminados.
- Compruebe que Todos y Sistema están incluidos en la lista con permisos Permitir inicio. Si no lo están, utilice el botón Agregar para agregarlos a la lista. Haga clic en Aceptar cuando la lista esté completa.
- Seleccione la ficha Aplicaciones, resalte su servidor, DCOMDemo_Svr.Class1, y a continuación haga clic en el botón Propiedades.
- Seleccione la ficha General, establezca el nivel de autenticación en Predeterminado y, a continuación, seleccione la ficha Ubicación. La única opción activada debería ser Ejecutar la aplicación en este equipo.
- Seleccione la ficha Seguridad y compruebe que los permisos Usar permisos de acceso predeterminados y Usar permisos de inicio predeterminados están activadas.
- Seleccione la ficha Identidad, active la opción El usuario inicial, haga clic en Aceptar para cerrar el cuadro de diálogo Propiedades del servidor y, a continuación, haga clic de nuevo en Aceptar para cerrar Dcomcnfg. Como puede ver, el servidor de prueba utiliza toda la configuración

predeterminada. Al implementar sus propios servidores, debería definir valores concretos para su aplicación. Toda la configuración personalizada tiene prioridad sobre los valores predeterminados.

Ahora puede probar su servidor. En el equipo cliente, inicie el cliente y, a continuación, haga clic en el botón Ejecutar. Debería ver un cuadro de mensaje que indica la hora del servidor.

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (Formato APA).

1. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
2. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- Numero de la Práctica.

PRACTICA 10.

II.- Nombre de la Práctica.

Creación de un Cliente COM

III.- Competencia(s) a desarrollar:

- Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model,
- Seleccionar el modelo de cómputo distribuido pertinente para una aplicación específica.
- Identificar las características, ventajas y desventajas del Modelo de Objetos de Componentes Distribuidos de Microsoft.
- Desarrollar aplicaciones bajo DCOM.
- Desarrollar aplicaciones bajo el Modelo de Objetos de Componentes Distribuidos de Microsoft (Distributed Component Object Model, DCOM), utilizando un lenguaje visual.

IV.- Introducción.

COM / DCOM

Microsoft Distributed COM (DCOM) extiende COM (Component Object Model) para soportar comunicación entre objetos en ordenadores distintos, en una LAN, WAN, o incluso en Internet. Con DCOM una aplicación puede ser distribuida en lugares que dan más sentido al cliente y a la aplicación.

Como DCOM es una evolución lógica de COM, se pueden utilizar los componentes creados en aplicaciones basadas en COM, y trasladarlas a entornos distribuidos. DCOM maneja detales muy bajos de protocolos de red, por lo que uno se puede centrar en la realidad de los negocios: proporcionar soluciones a clientes.

Actualmente DCOM viene con los sistemas operativos Windows 2000, NT, 98 y también está disponible una versión para Windows 95 en la página de Microsoft. También hay una implementación de DCOM para Apple Macintosh y se está trabajando en implementaciones para plataformas UNIX como Solaris.

La arquitectura DCOM

DCOM es una extensión de COM, y éste define como los componentes y sus clientes interactúan entre sí. Esta interacción es definida de tal manera que el cliente y el componente pueden conectar sin la necesidad de un sistema intermedio. El cliente llama a los métodos del componente sin tener que preocuparse de niveles más complejos.

En los actuales sistemas operativos, los procesos están separados unos de otros. Un cliente que necesita comunicarse con un componente en otro proceso no puede llamarlo directamente, y tendrá que utilizar alguna forma de comunicación entre procesos que proporcione el sistema operativo. COM proporciona este tipo de comunicación de una forma transparente: intercepta las llamadas del cliente y las reenvía al componente que está en otro proceso.

Cuando el cliente y el componente residen en distintas máquinas, DCOM simplemente reemplaza la comunicación entre procesos locales por un protocolo de red. Ni el cliente ni el componente se enteran de que la unión que los conecta es ahora un poco más grande.

Las librerías de COM proporcionan servicios orientados a objetos a los clientes y componentes, y utilizan RPC y un proveedor de seguridad para generar paquetes de red estándar que entienda el protocolo estándar de DCOM.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

4. - COM/DCOM (Component Object Model/ Distributed COM)

Subtemas:

4.1. Creación de Servidores COM.
4.2. Creación de un cliente COM.
4.3. Automatización.
4.4. ATL (Active Template Library).
4.5. DCOM (Distributed COM).

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.

IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red de local
- Software de Visual C#
- Manual de Visual C#

VIII.- Metodología.

Crear el cliente

- En el menú Archivo, seleccione la opción Nuevo proyecto, seleccione EXE estándar y, a continuación, haga clic en Aceptar. De forma predeterminada, se crea Form1.
- En el menú Proyecto, haga clic en la opción Propiedades del proyecto y, a continuación, seleccione la ficha General.
- En el campo Nombre de proyecto, escriba DCOMcliente.
- En el campo Descripción del proyecto, escriba DCOMCliente Proyecto - Cliente.
- En el menú Proyecto, haga clic en Referencias. En la lista de referencias disponibles, seleccione DCOMServer - Servidor.
- Coloque un botón de comando en Form1 y cambie el título del botón por Ejecutar.
- Coloque el código siguiente en el evento de clic del botón:

```
Dim MyObj As DCOMDemo_Svr.Class1

   On Error GoTo err1

   Set MyObj = CreateObject("DCOMDemo_Svr.Class1")
   MsgBox "Server Time=" & MyObj.ServerTime & "  Client
Time=" & Time

   Exit Sub
err1:
   MsgBox "Connection failed: Error " & Err.Number & " -
" & Err.Description
```

- En el menú Archivo, seleccione Guardar como y, a continuación, guarde el proyecto en la carpeta c:\DCOM\Cliente del cliente.
- Presione la tecla F5 para ejecutar el cliente en el IDE y probarlo.
- En el menú Archivo, seleccione Generar DCOMCliente para compilar el cliente y, a continuación, cierre Visual Basic.

Empaquetar el cliente

Al empaquetar el cliente, es necesario realizar algunos pasos concretos considerando que el servidor no se ejecuta en el mismo equipo que el cliente. Los cambios realizados al paquete del cliente garantizan que sólo se instala la biblioteca de tipos (archivo .tlb) y que se incluyen algunas entradas del Registro adicionales en lugar de la aplicación ejecutable del servidor, lo que no es necesario en el equipo del cliente porque no va a ejecutarse allí.

Use el procedimiento siguiente para empaquetar el cliente:

- Inicie el Asistente de empaquetado e implementación, y a continuación seleccione el proyecto del cliente.
- Haga clic en el botón Paquete. En el cuadro de diálogo Tipo de paquete, seleccione Paquete de instalación estándar y, a continuación, haga clic en Siguiente.
- En el cuadro de diálogo Carpeta de paquete, seleccione la carpeta donde almacenar el paquete y, a continuación, haga clic en Siguiente. En este caso, es c:\DCOM\Cliente\Package.

  NOTA: puede que aparezca un cuadro de diálogo que indique que no hay ninguna información de dependencia para el servidor. Haga clic en Aceptar porque este servidor no tiene ninguna dependencia.

  Debería estar ahora en el cuadro de diálogo Archivos incluidos.

- Anule la selección del archivo ejecutable del servidor, DCOMServer.exe, porque no desea distribuir la aplicación ejecutable del servidor y, a continuación, haga clic en el botón Agregar.
- Cambie el cuadro combinado Tipo a Archivos de servidor remoto (*.vbr).
- Señale a la carpeta donde tiene el proyecto del servidor (en este caso c:\DCOM\Servidor) y seleccione el archivo VBR relacionado, DCOMServer.VBR. Haga clic en Abrir y el cuadro de diálogo Agregar Archivo se cierra. Observe que se incluyen dos archivos, DCOMServer.VBR y DCOMServer.TLB. Haga clic en el botón Siguiente.

  NOTA: en el cuadro de diálogo Servidores remotos puede definir el nombre del equipo (dirección de red) donde el servidor se está ejecutando. Normalmente mantiene este campo en blanco porque puede no saber de antemano dónde se instalará el servidor. Si mantiene el espacio en blanco, se le preguntará la información al instalar el cliente. En este ejemplo, deje el espacio en blanco.

- Haga clic en Siguiente para continuar. Ahora puede proseguir con los procedimientos estándar para el Asistente de empaquetado e implementación. En este caso, basta con que haga clic en Siguiente en los demás cuadros de diálogo.

Instalar el cliente

Instale el cliente en el equipo en el que desearía ejecutarlo, utilizando el paquete de la distribución que creó anteriormente. Dado que este cliente utiliza un servidor DCOM y usted dejó la ubicación real del servidor en blanco cuando creó el paquete de distribución, ahora tiene que proporcionar esta ubicación. Cuando el programa de instalación la solicite, proporcione el nombre del equipo donde instaló el servidor.

IX.- Sugerencias didácticas.

- Siga correctamente los pasos indicados.
- Identifique los algoritmos
- Codifique los programas
- Ejecute los programas
- Valide su funcionamiento
- Compruebe con sus compañeros el funcionamiento.
- Investigar otras aplicaciones.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el

curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (Formato APA).

1. TORRES REMON, MANUEL. (2017). DESARROLLO DE APLICACIONES CON VISUAL BASIC 2015. Madrid, España: MARCOMBO.
2. Luna, Fernando O. (2011). Visual Basic. Buenos Airea: Fox Andina.
3. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
4. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 11.

II.- Nombre de la Práctica.

Crear un documento xml para un pedido.

III.- Competencia(s) a desarrollar:

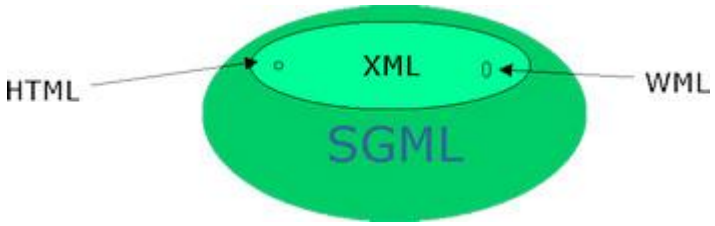
- Crear e implementar un servicio web.
- Comprender el funcionamiento de un servicio web.
- Desarrollar librerías de métodos remotos para realizar el intercambio de información estructurada en aplicaciones cliente-servidor.
- Publicar un servicio web para permitir su utilización por aplicaciones cliente.

IV.- Introducción.

XML ¿QUÉ ES?

El XML es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades. El XHTML, el MathML y el SVG son algunos de los lenguajes que el XML.

- XML es un subconjunto de SGML(Estándar Generalised Mark-up Language),simplificado y adaptado a Internet



- XML no es, como su nombre puede sugerir, un lenguaje de marcado.
- XML es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.

Que no es XML

- No es una versión mejorada de HTML
- HTML es una aplicación de SGML por lo tanto de XML
- No es un lenguaje para hacer páginas WEB
- Y sobre todo no es difícil

¿Por qué XML?

- Es un estándar internacionalmente conocido
- No pertenece a ninguna compañía

- Permite una utilización efectiva en Internet para sus diferentes terminales.

Definición

Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos.

¿Para qué sirve XML?

- Representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

Ventajas de XML

- Fácilmente procesable
- Separa radicalmente el contenido y el formato de presentación
- Diseñado para cualquier lenguaje y alfabeto. (encoding)

Características

- XML es un subconjunto de SGML que incorpora las tres características más importantes de este:
    - Extensibilidad
    - Estructura
    - Validación
- Basado en texto.
- Orientado a los contenidos no presentación.
- Las etiquetas se definen para crear los documentos, no tienen un significado preestablecido.
- No es sustituto de HTML.
- No existe un visor genérico de XML.

Aplicaciones de XML

- Publicar e intercambiar contenidos de bases de datos.
- Formatos de mensaje para comunicación entre aplicaciones (B2B)
- Descripción de meta contenidos.

Documento XML

- Conjunto de datos con sus respectivas etiquetas de marcado XML.
- Se almacena como texto en archivo con extensión .xml.
- Un documento XML puede incluir cualquier flujo de datos basado en texto: un artículo de una revista, un resumen de cotizaciones de bolsa, un conjunto de registros de una base de datos, etc..

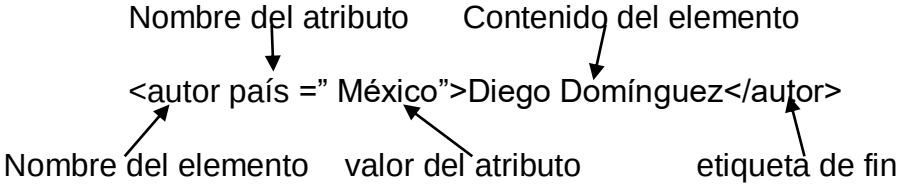
Estructura de un documento XML

- Un documento XML está formado por datos de caracteres y marcado, el marcado lo forman las etiquetas:

Prologo
<?xml versión =”1.0” encoding = “ISO-8859-1” standalone=”no”?>
<!DOCTYPE persona SYSTEM “persona.dtd”>

Cuerpo
<persona>
<nombre>Francisco</nombre>
<apellido>Martínez</apellidos>
<persona>

Estructura



Componentes de un documento XML

- En un documento XML existen los siguientes componentes:
  - Elementos: Pieza lógica del marcado, se representa con una cadena de texto(dato) encerrada entre etiquetas. Pueden existir elementos
    vacíos (
). Los elementos pueden contener atributos.
  - Instrucciones: Ordenes especiales para ser utilizadas por la aplicación que procesa
    <?xml-stylesheet type=“text/css” href=“estilo.css”>
  - Las instrucciones XML. ¿Comienzan por <? Y terminan por?>.
  - Comentarios: Información que no forma parte del documento. Comienzan por <!-- y terminan por -->.
  - Declaraciones de tipo: Especifican información acerca del documento:
    <!DOCTYPE persona SYSTEM “persona.dtd”>
  - Secciones CDATA: Se trata de un conjunto de caracteres que no deben ser interpretados por el procesador:
  - <![CDATA[Aquí se puede meter cualquier carácter, como <, &, >, ... Sin que sean interpretados como marcación]]>

Sintaxis de XML

- Representa las normas a seguir para la construcción de documentos XML.
- Estas reglas son dictadas por el organismo W3C (http://www.w3.org/XML). Entre ellas destacan:
- El XML es Case - Sensitive.
- Todo elemento tiene que tener su correspondiente etiqueta de inicio y de

cierre, o una sola etiqueta vacía.
- Todo documento, debe haber un elemento (llamado raíz de documento) que contenga a los demás.
- Todos los elementos deberán estar correctamente anidados.
- Todos los valores de los atributos deberán ir entre comillas.

Normas de buena construcción

- La primera letra del nombre se escribirá en mayúscula
- Los nombres compuestos se escribirán juntos o separados por guion bajo Saca_corchos
- Los elementos han de comenzar por un carácter o "_" no numérico

Normas de buena construcción II

- Existen 2 tipos de construcciones
  - Orientado a la presentación

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

5. Servicios web XML

Subtemas:

5.1. Características del lenguaje
5.2. Visión general de servicios web XML
5.3. Tecnologías subyacentes.
5.3.1. SOAP
5.3.2. WSDL
5.3.3. UDDI

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo

VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.
XV. Quitar protecciones de seguridad de los equipos.
XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.
XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.
XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red local
- Editor de texto.
- Navegador
- Manual de XML.
- Manual de MySQL.

VIII.- Metodología.

En esta práctica vamos a realizar lo siguiente:

- Cargar un archivo xml a mysql
- Generar un archivo xml a partir de una tabla de mysql
- Exportar consultas de una tabla de mysql a un archivo xml.

Cargar un archivo xml a mysql

Tenemos una tabla en mysql que contiene los siguientes datos:

1. Código
2. Nombre
3. Descripción
4. Valor

Tabla producto.

```
mysql> show tables;
+------------------+
| Tables_in_prueba |
+------------------+
| producto         |
+------------------+
1 row in set (0.00 sec)

mysql>
```

Con laSiguiente Estructura

```
mysql> describe producto;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_prod     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| codigo      | int(11)          | YES  |     | NULL    |                |
| nombre      | varchar(20)      | YES  |     | NULL    |                |
| descripcion | varchar(50)      | YES  |     | NULL    |                |
| precio      | float            | YES  |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Creamos el archivo xml y tenemos:

<?xml version="1.0" encoding="UTF-8"?>
<Productos>
<articulo id_prod="5" codigo="104" nombre="Monitor DELL" descripcion="Monitor Dell. 13 pulgadas" precio = “1200”/>
<articulo id_prod="6" codigo="105" nombre="Monitor HP" descripcion="Monitor HP. 17 pulgadas" precio = “1100”/>
<articulo id_prod="7" codigo="106" nombre="Monitor ACER" descripcion="Monitor Del7 pulgadas" precio = “1300”/>
<articulo id_prod="8" codigo="107" nombre="Monitor LENOVO" descripcion="Monitor LENOVO 15 pulgadas" precio = “1000”/>
</Productos>

Escribimos el siguiente código en mysql.

load xml local infile "c:/madrid/articulos.xml" into table producto rows identified by "<articulo>";
y tenemos:

```
mysql>
mysql> load xml local infile "c:/madrid/articulos.xml" into table producto rows identified by "<articulo>";
Query OK, 4 rows affected (0.06 sec)
Records: 4  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
```

Consultamos la table (select * from producto)

```
mysql> select * from producto;
+---------+--------+------------------+-----------------------------------------------------+--------+
| id_prod | codigo | nombre           | descripcion                                         | precio |
+---------+--------+------------------+-----------------------------------------------------+--------+
|       1 |    100 | Computadora DELL | Computadora Dell. Disco Duro 300 GB DDR 8 GB        |  12000 |
|       2 |    101 | Computadora EPSON| Computadora EPSON. Monitor 13 pulgadas DD 500 GB R  |  17000 |
|       3 |    101 | Computadora VAIO | Computadora VAIO. Monitor 13 pulgadas DD 200 GB RA  |  11000 |
|       4 |    103 | Computadora HP   | Computadora HP. Monitor 13 pulgadas DD 500 GB RAM   |  15000 |
|       5 |    104 | Monitor DELL     | Monitor Dell. 13 pulgadas                           |   1200 |
|       6 |    105 | Monitor HP       | Monitor HP. 17 pulgadas                             |   1100 |
|       7 |    106 | Monitor ACER     | Monitor De 17 pulgadas                              |   1300 |
|       8 |    107 | Monitor LENOVO   | Monitor LENOVO  15 pulgadas                         |   1000 |
+---------+--------+------------------+-----------------------------------------------------+--------+
8 rows in set (0.00 sec)
```

Y ya están insertados los datos en la tabla

Generar un archivo xml a partir de una tabla de mysql

Escribir en el prompt del sistema operativo lo siguiente:
*mysql --xml -emysql "select * from prueba.producto" > persona.xml;*

Exportar consultas de una tabla de mysql a un archivo xml.

Tenemos una tabla en mysql que contiene los siguientes datos:

5. Id del producto
6. Código
7. Nombre
8. Descripción
9. Valor

Tabla producto.

```
mysql> show tables;
+-----------------+
| Tables_in_prueba |
+-----------------+
| producto         |
+-----------------+
1 row in set (0.00 sec)

mysql>
```

Con laSiguiente Estructura

```
mysql> describe producto;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_prod     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| codigo      | int(11)          | YES  |     | NULL    |                |
| nombre      | varchar(20)      | YES  |     | NULL    |                |
| descripcion | varchar(50)      | YES  |     | NULL    |                |
| precio      | float            | YES  |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

Con el siguiente Contenido

```
mysql> SELECT * FROM PRODUCTO;
+---------+--------+------------------+-----------------------------------------------+--------+
| id_prod | codigo | nombre           | descripcion                                   | precio |
+---------+--------+------------------+-----------------------------------------------+--------+
|       1 |    100 | Computadora DELL | Computadora Dell. Disco Duro 300 GB DDR 8 GB  |  12000 |
|       2 |    101 | Computadora EPSON| Computadora EPSON. Monitor 13 pulgadas DD 500 GB R |  17000 |
|       3 |    101 | Computadora VAIO | Computadora VAIO. Monitor 13 pulgadas DD 200 GB RA |  11000 |
|       4 |    103 | Computadora HP   | Computadora HP. Monitor 13 pulgadas DD 500 GB RAM |  15000 |
+---------+--------+------------------+-----------------------------------------------+--------+
4 rows in set (0.00 sec)
```

Escribimos el siguiente Codigo en mysql

```
mysql> select concat("<Producto><Código>",codigo,
"</Codigo><Nombre>",nombre,"
</Nombre><Descripcion>",descripcion,"</Descripcion><Precio>",precio,
"</Precio></Producto>") as "<Datos>" from producto into outfile
"c:\madrid\productos.xml";
```

Y nos notifica que ya se genero la consulta en el archivo

```
mysql> select concat("<Producto><Codigo>",codigo, "</Codigo><Nombre>",nombre," </Nombre><Descripcion>",descripcion,"</Descripci
on><Precio>",precio, "</Precio></Producto>") as "<Datos>" from producto into outfile "c:\madrid\productos.xml";
Query OK, 4 rows affected (0.00 sec)

mysql>
```

Y vemos el archivo XML

```
C:\madrid>dir *.xml
 Volume in drive C is OS
 Volume Serial Number is 6EDB-0B0A

 Directory of C:\madrid

25/10/2018  01:02 p.m.               693 productos.xml
               1 File(s)            693 bytes
               0 Dir(s)  104,195,735,552 bytes free

C:\madrid>
```

Y tenemos el archivo XML

```
<Producto>
    <Código>100</Codigo>
    <Nombre>Computadora DELL </Nombre>
    <Descripcion>Computadora Dell. Disco Duro 300 GB DDR 8 GB</Descripcion>
    <Precio>12000
    </Precio>
</Producto>
<Producto>
    <Código>101</Codigo>
```

```
    <Nombre>Computadora EPSON </Nombre>
    <Descripcion>Computadora EPSON. Monitor 13 pulgadas DD 500 GBR
    </Descripcion>
    <Precio>17000</Precio>
</Producto>
<Producto>
   <Código>101</Codigo>
   <Nombre>Computadora VAIO </Nombre>
   <Descripcion>Computadora VAIO. Monitor 13 pulgadas DD 200 GB RA
   </Descripcion><Precio>11000</Precio>
</Producto>
<Producto>
    <Código>103</Codigo>
    <Nombre>Computadora HP </Nombre>
    <Descripcion>Computadora HP. Monitor 13 pulgadas DD 500 GB RAM
    </Descripcion>
    <Precio>15000</Precio>
</Producto>


<?xml version="1.0" encoding="utf-8"?>
<Raiz>
  <Clientes>
    <Cliente ClienteID="101">
      <Empresa>Surtidora de Alimentos Sonora </Empresa>
      <Contacto>Honoracio Gutierrez</Contacto>
      <Puesto_contacto>Gerente de Ventas</Puesto_contacto>
      <Telefono>(01)555-7555</Telefono>
      <Dirección>
        <Domicilio>2732 Blvd Capomo.</Domicilio>
        <Ciudad>Hermosillo</Ciudad>
        <Estado>SONORA</Estado>
        <CodigoPostal>97403</CodigoPostal>
        <Pais>MÉXICO</Pais>
      </Dirección>
    </Cliente>
    <Cliente ClienteID="102">
      <Empresa>Importaciones Coyote</Empresa>
      <Contacto>Yolanda Lomelí</Contacto>
      <Puesto_contacto>Representante de Ventas</Puesto_contacto>
      <Telefono>(01)555-6874</Telefono>
      <Fax>(01)555-2376</Fax>
      <Dirección>
        <Domicilio>Calle de la Reforma 34</Domicilio>
        <Ciudad>Obregon</Ciudad>
        <Estado>SONORA</Estado>
```

```
      <CodigoPostal>97827</CodigoPostal>
      <Pais>MÉXICO</Pais>
    </Dirección>
  </Cliente>
  <Cliente ClienteID="103">
    <Empresa>La Mejor de Alimentos</Empresa>
    <Contacto>Juan Soria</Contacto>
    <Puesto_contacto>Gerente de Ventas</Puesto_contacto>
    <Telefono>(01) 555-7969</Telefono>
    <Fax>(01) 555-6221</Fax>
    <Dirección>
      <Domicilio>12 Ave. Monteverde 244</Domicilio>
      <Ciudad>Nogales</Ciudad>
      <Estado>SONORA</Estado>
      <CodigoPostal>99362</CodigoPostal>
      <Pais>MÉXICO</Pais>
    </Dirección>
  </Cliente>
  <Cliente ClienteID="104">
    <Empresa>Let's Stop N Shop</Empresa>
    <Contacto>Jaime Sonorares</Contacto>
    <Puesto_contacto>Propietario</Puesto_contacto>
    <Telefono>(415) 555-5938</Telefono>
    <Dirección>
      <Domicilio>87 Polk St. Suite 5</Domicilio>
      <Ciudad>San Francisco</Ciudad>
      <Estado>California</Estado>
      <CodigoPostal>94117</CodigoPostal>
      <Pais>EU</Pais>
    </Dirección>
  </Cliente>
</Clientes>
```

IX.- Sugerencias didácticas.

- Identifique los pasos para elaborar el programa
- Escriba los pasos en un editor de texto.
- Valide que sea un documento bien formado.
- Ejecútelo con el Navegador
- Valide los resultados.
- Compruebe con sus compañeros el funcionamiento.
- Investigar otros ejemplos.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (formato APA).

1. Thierry BOULANGER. (2015). XML práctico Bases esenciales, conceptos y casos prácticos (2ª edición). Madrid, España: Ediciones ENI.
2. W3C (James Clark, Steve DeRose). (2017). Lenguaje de Caminos XML (XPath). Buenos Aires: Grupo Sidar.
3. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
4. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.
5. Wong, Paul. Java. Ed. International Thomsom Editores.
6. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
7. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
8. Froute, Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.

I.- No. De la Práctica.

PRACTICA 12.

II.- Nombre de la Práctica.

Servicio Web

III.- Competencia(s) a desarrollar:

- Comprender el funcionamiento de un servicio web.
- Desarrollar librerías de métodos remotos para realizar el intercambio de información estructurada en aplicaciones cliente-servidor.
- Publicar un servicio web para permitir su utilización por aplicaciones cliente.

IV.- Introducción.

Creación de Un WebService en Java

Un servicio web ***(en inglés, Web service)*** es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre programas. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de computadoras como Internet.

Los Web services representan la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente en las cuales el software está distribuido en diferentes servidores; de esta manera se logra la creación de grandes aplicaciones que pueden funcionar en una gran multitud de dispositivos, entre los que se encuentran los teléfonos móviles, las tabletas y computadores personales, todos ellos interactuando con un servidor.

Para comenzar la práctica de cómo se crean y utilizan los webservices, es necesario considerar tener claros algunos conceptos, como son:

WSDL

En ocasiones leído como como wisdel son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo.

SOAP (Simple Object Access Protocol)

El protocolo estándar que se utiliza para enviar la información, es SOAP. Este define el formato del “envelope” que se intercambia entre cliente y servicio, así como las convenciones para representar invocaciones y respuestas Estos mensajes son transmitidos en formato XML, montado sobre HTTP.

Bien, esto es lo básico que hay que tener claro para comenzar a crear un web service, existe mucha teoría e información que se podría consultar de ser necesario, pero en esta práctica solo hay que tener presente lo que hemos citado.

V.- Correlación con los temas y subtemas del programa de estudio.

Temas:

5. Servicios web XML

Subtemas:

- 5.1. Características del lenguaje
- 5.2. Visión general de servicios web XML
- 5.3. Tecnologías subyacentes.
  - 5.3.1. SOAP
  - 5.3.2. WSDL
  - 5.3.3. UDDI

VI. Medidas de seguridad e higiene

SEGURIDAD E HIGIENE SOBRE EL AREA DE TRABAJO (Laboratorio de Computo)

I. NO Fumar, introducir y/o consumir alimentos o bebidas en los espacios del Laboratorio.
II. Presentarse bajo los efectos de bebidas embriagantes, narcóticos, drogas, enervantes o psicotrópicos, en los espacios del Laboratorio.
III. Introducir armas de fuego, punzo cortantes o de cualquier otro tipo, que pudieran poner en peligro la vida o salud de los usuarios.
IV. Hacer ruido excesivo.
V. Tirar basura.
VI. Sentarse en las escaleras, así como quedarse parado en las puertas, ya que se obstruye el paso.
VII. Sentarse en las mesas de trabajo
VIII. Correr dentro del Laboratorio.
IX. Desconectar o cambiar de lugar los elementos o periféricos conectados a las computadoras; tales como, teclados, ratones, etc.
X. El uso de mensajeros instantáneos (chats).
XI. El uso de cualquier tipo de juegos electrónicos.
XII. El uso, instalación, revisión y acceso a sitios de pornografía.
XIII. La instalación de software que no sea avalado por el encargado del Laboratorio.
XIV. Dañar intencionalmente cualquier componente del equipo de cómputo, así

como extraer, borrar o cambiar la forma de operación del programa instalado en la computadora. Los usuarios que dañen los programas o equipo deberán pagar el costo de reparación o de adquisición según sea el caso, independientemente de la sanción a que se hagan acreedores.

XV. Quitar protecciones de seguridad de los equipos.

XVI. La permanencia de alumnos en las aulas didácticas cuando el(la) profesor(a) del grupo o el encargado de Laboratorio no esté presente, salvo que el usuario pida autorización y se haga responsable del equipo.

XVII. La permanencia de alumnos en las instalaciones del Laboratorio en espera del profesor toda vez que ello propicia la alteración del orden.

XVIII. Alterar en cualquier forma, los métodos, procedimientos y sistemas establecidos.

VII.- Material y equipo necesario.

- Computadora.
- Red local
- Editor de texto.
- Navegador.
- Manual de XML.

VIII.- Metodología.

Para facilitar más la comprensión de la práctica que vamos a realizar diremos que (en términos "coloquiales" de programación) un webservice es aquel programa que está comprendido por funciones o métodos que reciben parámetros o valores, los procesan y arrojan una salida.

En esta práctica vamos a utilizar netbeans y el servidor apache tomcat que viene con dicha versión de netbeans. Los pasos a seguir son los siguientes:

- Primero, creamos un nuevo proyecto de netbeans, de tipo java web Application.
- Luego, definimos el nombre de nuestro proyecto:
- Ahora presionamos el botón siguiente y escogemos el servidor Apache Tomcat
- Volvemos a dar clic sobre el botón siguiente y en la ventana que aparece hacemos clic en terminar sin elegir ningun framework:
- Listo ahora debe de aparecernos una ventana que corresponde a la página index del proyecto (código html)

Sin embargo, dicha página no será mostrada ya que no será necesario visualizar ninguna información html, todo el proceso será codificado en otro lugar.

- Par continuar hacemos clic derecho sobre nuestro Proyecto (Ejemplo_Servicio_web) y en la lista desplegable que aparece seleccionamos la opción nueva y en la sublista escogemos Paquete Java.
- Aparece nuestro paquete que almacenara el código javo de nuestro webservice:
- Ahora sí, podemos crear el webservice como tal, para ello hacemos nuevamente clic derecho sobre el proyecto y en la lista desplegable que aparece, escogemos la opción nueva y en la sublista Web Service.
- Se abrirá una nueva ventana donde le colocaremos nombre a nuestro web service (ejem. proceso_wc) y seleccionamos el paquete que antes creamos:
- Listo, ya está creado nuestro web service (vacio).
- Bien, hasta este punto se han explicado los pasos para la creación de un webservice en netbeans, pero aún no se ha mencionado para que queremos usar el webservice; Entonces diremos que nuestro webservice se usará para registrar usuarios en una base de datos, gestionar los datos de dichos usuarios almacenados con procesos de actualización, eliminación y búsquedas. Así que surge la necesidad de usar el conector de MySql para java, dicho conector debe de ir en la carpeta Web-Inf del Proyecto.
- Bien, teniendo alojado el conector de mysql en la carpeta correcta del proyecto se puede pasar a agregar el jar como referencia de la siguiente manera, hacer clic derecho sobre el proyecto y escoger la opción propiedades:
- Después, en la siguiente ventana que aparece se presiona el botón Add Jar/folder y en la nueva ventana que aparece buscamos la librería en el directorio de proyectos de netbeans.
- Teniendo ubicada la librería se procede a hacer clic sobre el botón abrir.
- Bien, Ahora es necesario definir la base de datos y la tabla que vamos a usar para almacenar la información.
- Abrimos nuestro administrador de bases de datos y creamos una base de datos llamada (datos_almacenados_ws) y luego dentro de ella una tabla llamada usuarios con los siguientes campos campos:

  - ✓ Clave, rfc, nombre, apellido, sexo.

- Listo, ya está definida nuestra tabla, ahora si podemos pasar directamente a codificar en Java nuestro Web Service.
- Estando en nuestro proyecto en Netbeans, Importamos las siguientes librerías en el archivo llamado proceso_wc:

```
import javax.jws.WebService;
import java.io.*;
import java.sql.*;
import javax.jws.WebMethod;
import javax.swing.JOptionPane;
```

- Luego dentro de la clase proceso_wc debemos declarar las siguientes variables:

```
@WebService()
public class proceso_wc {
private int id;
private String cedula;
private String nombre;
private String apellido;
private String direccion;
private String sexo;
// Ojo, aqui van las funciones que usara el webservice para realizar los diferentes procesos con los datos
}
```

- Bien Ahora si definamos los métodos o funciones que constituirán nuestro Web Service, comencemos con insertar Usuario:

```
@WebMethod
public boolean insertarUsuario(String ce,String no,String ap, String dir, String se)
String
conexionBD="jdbc:mysql://127.0.0.1/datos_almacenados_ws";
Connection conexion=null;
boolean funciono=false;
String con;
try{
// JOptionPane.showMessageDialog(null,"entro");
Class.forName("com.mysql.jdbc.Driver");//el driver de mysql
conexion=DriverManager.getConnection(conexionBD,
"root","Madrid1");//conexion a la base de datos
Statement s = conexion.createStatement();
// JOptionPane.showMessageDialog(null,no+" "+ap);
con= "INSERT INTO USUARIOS VALUES (NULL,'"+ce+"','"+no+"','"
+ ap +"','"+ dir +"','"+se+"')";
s.executeUpdate(con);
funciono=true;
}
catch(Exception e){
System.out.println("No se ha completado la petición...");
}
return funciono;
}
```

- Ahora pasemos a desarrollar el procedimiento de búsqueda, el cual recibe como parámetro la cedula de un usuario y luego a través de una

consulta sql se determina si el usuario existe, dicha función se llama buscarUsuarioCedula y se muestra a continuación:

```
@WebMethod
public boolean buscarUsuarioCedula(String ced){
String
conexionBD="jdbc:mysql://127.0.0.1/datos_almacenados_ws";
Connection conexion=null;
boolean funciono=false
String con;
ResultSet rs;
try{
// JOptionPane.showMessageDialog(null,"entro");
Class.forName("com.mysql.jdbc.Driver");//el driver de mysql
conexion=DriverManager.getConnection(conexionBD,
"root","12345");//conexion a la base de datos
Statement s = conexion.createStatement();
// JOptionPane.showMessageDialog(null,no+" "+ap);
con="SELECT * FROM usuarios where cedula = '" + ced + "'" ;
rs = s.executeQuery (con); {
while (rs.next()) {
cedula=rs.getString("cedula");
nombre=rs.getString("nombre");
apellido=rs.getString("apellido");
direccion=rs.getString("direccion");
sexo=rs.getString("sexo");
funciono=true;
MostrarCedula();
MostrarNombre();
MostrarApellido();
MostrarDireccion();
MostrarSexo();
break;
//JOptionPane.showMessageDialog(null, "si");
}
}
catch(Exception e){
System.out.println("No se ha completado la peticiÃ³n...");
}
return funciono;
}
```

- Como se Puede ver, la función buscarUsuarioCedula utiliza varias subfunciones para permitir la visualización de los datos como son, MostrarCedula(), MostrarNombre(), MostrarApellido(), MostrarDireccion(), MostrarSexo().
- A continuación, Definirá cada función para que retorne el valor correspondiente:

```
@WebMethod
public String MostrarNombre(){
String nomb;
nomb="";
nomb=nombre;
return nomb;
}
@WebMethod
public String MostrarApellido(){
String apell;
apell="";
apell=apellido;
return apell;
}
@WebMethod
public String MostrarCedula(){
String codi;
codi="";
codi=cedula;
return codi;
}
@WebMethod
public String MostrarDireccion(){
String dire;
dire="";
dire=direccion;
return dire;
}
@WebMethod
public String MostrarSexo(){
String se;
se="";
se=sexo;
return se;
}
```

- Ahora desarrolle el procedimiento para actualizar usuario, como se ha notado, cada función recibe parámetros y retorna un valor para que otra aplicación que interactúe con el web service determine si el proceso se realizó de forma correcta.
- Elabore el procedimiento para eliminar al usuario.
- Bien, Hasta aquí ya está listo nuestro servicio web, llego la hora de probarlo, para ello debemos hacer lo siguiente:

    - ✓ Presionar clic derecho sobre el proyecto Ejemplo_servicio_web y en el menú desplegable que aparece seleccionamos la opcion Limpiar y construir.

- Luego volvemos a presionar clic derecho sobre el proyecto y seleccionamos la opción deploy para que el proyecto se despliegue, es decir se compile y se coloque en funcionamiento:
- A continuación, nos dirigimos a la carpeta webservice del directorio de nuestro proyecto y presionamos clic derecho sobre el archivo proceso_wc y en la lista desplegable que aparece seleccionamos la opción Test Web service.
- Listo de esa manera se pone en marcha nuestro web service en el navegador.
- Allí podemos observar la WSDL de nuestro servicio web, esa dirección es la que necesitamos para poder interactuar desde otra aplicación (realizada en cualquier lenguaje y dispositivo) con nuestro Web Service.

IX.- Sugerencias didácticas.

- Identifique los pasos para elaborar el programa
- Escriba los pasos en un editor de texto.
- Valide que sea un documento bien formado.
- Ejecútelo con el Navegador
- Valide los resultados.
- Compruebe con sus compañeros el funcionamiento.
- Investigar otros ejemplos.

X.- Reporte del alumno.

La práctica desarrollada por el alumno será revisada en el salón de clase y esta deberá subirse a la página de cursos en línea del ITH en ***materias.ith.mx*** en el curso algoritmos y lenguajes de programación en el apartado de tareas con el título de practica base de datos. Esta tarea deberá contener en un link a un archivo el siguiente texto en letra arial 12:

- Nombre del alumno
- Grupo (salón)
- Hora

XI.- Bibliografía (formato APA).

1. Vukotic, Aleksa. Goodwill. (2017), James. Apache Tomcat 7. La voz expert en Java. Apress. Friendsof.
2. Decker, Hirshfield. (2001). Programación con Java. 2ª. Edición. México: International Thomsom Editores.
3. Jesús Bobadilla (2003). Java a través de ejemplos. México: Alfa Omega – RAMA.
4. Kris Jamsa.(1999). Aprenda y practique Java.Ed. Oxford.

5. Wong, Paul. Java. Ed. International Thomsom Editores.
6. Márquez, Francisco M. Unix Programación Avanzada 2ª. Edición. México: Alfaomega Ra-Ma.
7. Ryan, Timothy W. Distributed Object Technology. Prentice Hall.
8. Froute , Agustín. Java 2 Manual de Usuario y Tutorial 2ª. edición. Alfaomega Rama.
9. Candela, Santiago. García, Rubén. Quesada, Alexis. Santana, Francisco. Santos, Miguel. (2016), Fundamentos de sistemas operativos. Madrid. Thomson Editores Spain.
10. Martínez Acosta, Deivis de Jesús. (2014), Herramienta para la generación del código fuente para aplicaciones con arquitectura Modelo Vista Controlador (MVC) bajo desarrollo dirigido por modelos textuales (MDD). Msc. en Ingeniería de Sistemas y Computación. 1 Volumen, Bogotá D.C.: Universidad Nacional de Colombia.
11. Martin Kalin. (2009). introduction to JAX-WS (Java API for XML-Web Services). 15-octubre-2016, de O'Reilly Media, Inc. Sitio web: https://swrdfish.github.io/assets/ssl/JavaWebServices.pdf
12. Víctor J. Sosa Sosa. (marzo 10, 2015). Servicios Web: SOAP y REST. Octubre 2017, de Cinvestav-Tamaulipas Sitio web: https://www.tamps.cinvestav.mx/~vjsosa/clases/sd/ServiciosWeb_Axis_REST.pdf