Fundamento Bases de Datos
Facultad de Ciencias UNAM
M.I. Gerardo Avilés Rosas
gar@ciencias.unam.mx
Laboratorio:
Efraín Hipólito Chamú <*chamugauss@gmail.com* >

PRACTICA 02
Sistemas de Archivos

1. Introducción

Un sistema de archivos es un conjunto de normas y procedimientos para almacenar la información. Todo sistema operativo tiene uno y es la manera en que prefiere guardar la información. Los sistemas de archivos proveen métodos para crear, mover, renombrar y eliminar tanto archivos como directorios.

Generalmente un sistema de archivos tiene directorios que asocian nombres de archivos con archivos, usualmente conectando el nombre de archivo a un índice en una tabla de asignación archivos de algún tipo, como FAT en sistemas de archivos MS-DOS o los inodos de los sistemas Unix. La estructura de directorios puede ser plana o jerárquica (ramificada o "en árbol"). En algunos sistemas de archivos los nombres de archivos son estructurados, con sintaxis especiales para extensiones de archivos y números de versión. En otros, los nombres de archivos son simplemente cadenas de texto y los metadatos de cada archivo son alojados separadamente.

Los sistemas de archivos pueden ser clasificados en tres ramas:

- Sistemas de archivos de disco:

Un sistema de archivo de disco está diseñado para el almacenamiento de archivos en una unidad de disco, que puede estar conectada directa o indirectamente a la computadora.

- Sistemas de archivos de red:

Los sistemas de archivo de red permiten acceder a ficheros remotos como si estuviesen en un medio de almacenamiento local:

- Sistemas de archivos de propósito especial:

Los Sistema de archivos de propósito especial (en inglés Special purpose file system) son aquellos tipos de sistemas de archivos que no son ni sistemas de archivos de disco, ni sistemas de archivos de red.

Ejemplos: acme (Plan 9), archfs, cdfs, cfs, devfs, udev, ftpfs, lnfs, nntpfs, plumber (Plan 9), procfs, ROMFS, swap, sysfs, TMPFS, wikifs, LUFS, etc.

A. Manejo de Archivos en Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*").

En Java existen dos tipos de manejo de archivos, para escritura y lectura.

Para escritura de Archivos tenemos dos clases importantes:

I. FileWriter: Se encarga de crear un archivo y si no existe lo crea.
II. BufferedWriter: Sirve para escribir en un archivo, recibe como parámetro un objeto de tipo FileWriter, para escribir se utiliza el método write().

Para lectura de un Archivo utilizamos:

I. FileReader: Esta clase tiene métodos que nos permite leer caracteres, para ello se utiliza el método read().
II. BufferedReader: Utilizada para leer líneas completas de un archivo utilizando el método readLine().

Para borrar un archivo debemos invocar el método delete() de la clase File.

B. Ejemplo

```java
public class Hercules {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        try {
            //Instanciamos el objeto File para después crearlo.
            File cl = new File("clientes.txt");

            //Creamos el archivo clientes.txt
            FileWriter clw = new FileWriter(cl);

            //Escribimos el cliente1 en el archivo.
            clw.write("cliente1");

            //Cerramos el archivo.
            clw.close();

            //Usando BufferedWriter.
            FileWriter fw = new FileWriter("clientes_gym.txt");
            BufferedWriter clb = new BufferedWriter(fw);

            //Escribiendo en el archivo.
            clb.write("dato1");
            clb.newLine();
            clb.write("dato2");

            //Guardamos cambios en el archivo clientes_gym.txt
            clb.flush();

            //Leemos el contenido del archivo para después imprimirlo.
            FileReader fr = new FileReader(cl);
            BufferedReader br = new BufferedReader(fr);

            String lineaCliente = br.readLine();
            System.out.println(lineaCliente);

        } catch (IOException e) {
            System.err.printf("Error:", e);
        }
    }

}
```

C. Análisis de requerimientos

El proceso de generar soluciones a problemas planteados por una persona u organización requiere el diseño de forma adecuada, siempre y cuando se tenga la información correcta. El generar procesos automáticos es una forma que se aprende a lo largo del tiempo, la cuestión es saber ¿qué modelar o que codificar?.

Para saber que necesita la empresa o el cliente, es necesario entrevistarse con él, saber sus prioridades y conocer el problema real a resolver, así tendremos herramientas para poder atacar el problema y aquellos detalles que necesitamos saber para tomar la mejor solución.

En consecuencia, un Diseñador de Bases de Datos (DBD), muchas veces es un agente externo que pretende dar soluciones al cliente, generando modelos a los procesos ya establecidos. Por lo que el DBD se enfrentará a dos grandes problemas:

- ➢ El DBD ignora cuáles son las expectativas del nuevo sistema, es decir, desconoce los procesos de la empresa que se pretenden automatizar.
- ➢ El DBD al no ser el cliente final del sistema, no recibe retroalimentación alguna de este y por lo tanto no puede ajustar su modelo al mismo tiempo que lo prueba, es decir, no puede aplicarse el método de ensayo y error.

Para satisfacer las necesidades del cliente es necesario conocer y analizar los requerimientos en cuatro rubros:

1) Enumerar los requerimientos candidato
   En esta actividad se identificarán las necesidades expresadas por la organización, mismas que serán tratadas como requerimientos de alto nivel, es decir, aún no se especificarán a detalle. La lista que resulta de esta actividad tiene una estructura que permite, posteriormente, añadir diferentes tipos de información con el fin de proveer un panorama más amplio de las cualidades de cada requerimiento.

2) Comprensión del contexto del sistema
   Para lograr una comprensión del contexto del sistema que se pretende modelar, es necesario conseguir un entendimiento global del problema, es decir, encontrar todas las relaciones posibles entre los diferentes actores del sistema (trabajadores, clientes, proveedores, etc.) y las actividades que cada uno debe realizar.
   Esta información se puede representar mediante un diagrama que modele la acción de cada actor, estableciendo así un modelo del flujo de trabajo de la organización en cuestión.

3) Captura de requerimientos funcionales
   Los requerimientos funcionales son aquellas características o acciones que definen de manera única el flujo de trabajo de la organización.

4) Captura de requerimientos no funcionales
   Los requerimientos no funcionales pueden ser clasificados en dos diferentes grupos:
   - ➔ Requerimientos no funcionales asociados a requerimientos funcionales:
     Son requerimientos o reglas de negocio (por ejemplo: políticas de la empresa) que definen cada acción del flujo de trabajo, es decir, características propias de cada requerimiento funcional.
   - ➔ Requerimientos no funcionales no asociados a requerimientos funcionales:
     Estos requerimientos no dependen de alguna actividad específica del flujo de trabajo, es decir, que no detallan o describen el sistema. Sin embargo, son

requerimientos que deben tomarse en cuenta para la implementación del nuevo sistema, como por ejemplo: que plataforma usar, los recursos con los que se cuentan, etc.

2. Ejercicios

El gimnasio “Hércules” cuenta con área de pesas, cardio y dos salones donde se realizan diferentes actividades como Zumba, Twerk, Boxeo, Danza Árabe, Yoga, entre otras. Cuenta con regaderas, sauna y casilleros, además de una tienda donde se puede adquirir distintos artículos deportivos, suplementos alimenticios y bebidas.

Al ir aumentando la calidad del servicio “Hércules” planea abrir algunas sucursales, ya que ha generado mayor popularidad en los últimos años ocasionando que haya un incremento en el número de clientes, por lo cual requiere una mejora en el registro de entrada y salida.

Un problema frecuente es que para entrar en el gimnasio hay que anotarse en una libreta, en la cual se especifica su número de socio y las áreas que visitará provocando abuso de confianza por parte de los socios, ocasionando que asistan a clases que no tienen asignadas o que no pagaron. Otro inconveniente es que, dentro de la tienda de suplementos, la mala administración ha ocasionado escasez o exceso de productos.

Para esta práctica realiza lo siguiente:

- ❏ Crea un programa en Java que se acerque a la mejor solución para llevar el control del gimnasio, el programa deberá tener un menú donde se permitirá crear, eliminar y actualizar tanto clientes, clases impartidas y productos de la tienda, así como una opción de buscar, ésta permitirá buscar dentro de un archivo específico lo que se le pase como parámetro.
- ❏ Según sea requiera, crea los métodos necesarios para lo que se pide: crearCliente(), creaClasesaImpartir(), actualizarCliente(), actualizarClaseaImpartir(), eliminarCliente(), eliminarClase(), buscarCliente(), buscarClase, etc, de acuerdo a la lógica planteada a continuación.

  Nota: Los métodos no necesariamente deben de ir sin parámetros.

Para resolver el problema se tiene la siguiente lógica.

- ❏ Clientes:
  - ★ Crea el archivo correspondiente “Clientes.txt”, donde se podrá añadir un nuevo cliente sin perder los anteriores, siempre y cuando no exista. Deberá incluir nombre completo, dirección y teléfono, así como asignar un valor que permita identificarlo de manera única.

Ejemplo: 1 Efraín Hipólito Chamú Dirección1 Tel:5522222222.

- ★ Se deberá poder actualizar el archivo “Clientes.txt”, siempre y cuando exista el cliente.
- ★ No se podrá eliminar el archivo “Clientes.txt”, sin embargo, si se podrá eliminar un cliente específico del archivo.

❏ Clases:

- ★ Crea el archivo “Clases.txt” donde se deberá incluir el nombre de la clase, nombre del profesor que la imparte, horario de clase, días que se imparten y asignándole un identificador único.
- ★ Se deberá poder actualizar la lista de Clases, siempre y cuando exista la clase.
- ★ Se podrá eliminar clases, pero no se podrá eliminar el archivo.

❏ Clases impartidas:

- ★ Crea el archivo “ClasesImpartidas.txt” donde se deberá incluir los clientes que toman clases en el gimnasio, este deberá incluir el identificador del cliente e identificador de la clase impartida, tanto el cliente como la clase deberán existir.
- ★ Se deberá poder ingresar tantas clases impartidas como se soliciten, sin borrar las anteriores.
- ★ No se podrá eliminar el archivo ni las clases impartidas.

❏ Productos:

- ★ Crea el archivo “Productos.txt” donde se almacenarán todos los productos que hay para ofrecer a los clientes, deberá incluir: marca, nombre del producto, costo, descripción, cantidad de producto en existencia, presentación e identificador único.
- ★ Se podrá actualizar el nombre del producto y/o costo.
- ★ Se podrá ingresar nuevos productos, sin perder la información de los productos ya existentes.
- ★ No se podrá eliminar el archivo “Productos.txt”, pero si se podrá eliminar un producto de la lista.

❏ Productos vendidos:

- ★ Crea un archivo “ProductosVendidos.txt”, en el cual se podrá agregar el identificador del cliente e identificador de producto que compró, el cliente debe existir, así como el producto.
- ★ Se podrá ingresar tantos productos vendidos como sea posible.
- ★ No se podrá eliminar los productos vendidos ni el archivo.

❏ El método buscarEnEntidad() deberá realizar una búsqueda de un cliente, sobre Clientes.txt, ClasesImpartidas.txt y ProductosVendidos.txt. En caso de existir registro

del cliente se deberá crear el archivo “Reporte_Cliente_N.txt” donde N es el identificador del cliente y deberá contener las clases que está tomando, así como los productos que ha comprado. Si el cliente no existe indicar “Cliente no existe”.

Entregables

- Entrega tu proyecto o tus archivos .java acompañados de un archivo Clave.txt que incluya la forma en que deben ejecutarse (de preferencia en proyecto de NetBeans) siguiendo los lineamientos de entrega de prácticas.
- Generar un reporte llamado “Analisis_Requerimientos” en formato PDF en el que incluyas el Análisis de Requerimientos del gimnasio “Hércules”. Describe las ventajas y desventajas de usar un Sistema de Archivos para este tipo de problemas, así como las ventajas y desventajas de usar una Base de Datos.
- Si tuviste algún inconveniente, genera otro reporte en formato PDF llamado “Problemas”, mencionando los problemas a los que te enfrentaste al momento de solucionar la práctica.
- **Fecha de entrega domingo 27 de agosto a las 23:59 hrs.**

Notas finales

- Puedes crear el número de clases y métodos que creas necesarios/as.
- Comenta el código, para cada clase y método que usaste.