**TECNOLÓGICO DE COSTA RICA**

**SEDE REGIONAL SAN CARLOS.**

##### CARRERA:

INGENIERÍA EN COMPUTACIÓN

##### CURSO:

ALGORITMOS Y ESTRUCTURAS DE DATOS 1

##### TEC@WORKSHOP-MANAGER.PLUS

##### PROFESORA:

LORENA VALERIO SOLÍS.

**ELABORADO POR:**

LUIS ALONSO VEGA BRENES

YÁSER AUGUSTO CASTILLO BRAVO

***SANTA CLARA, NOVIEMBRE DE 2010.***

Descripción del problema

Se necesita un programa que pueda manejar ciertos datos de un taller mecánico. Los datos son los siguientes:

* Clientes
* Autos
* Empleados

Estos datos están relacionados entre sí. Cada cliente tiene uno o más autos, y a cada empleado se le puede asignar un trabajo que a su vez se relaciona con uno de los autos. Cada lista debe mostrarse (imprimirse al usuario) en un orden distinto, ya sea pre-orden, en orden o en post-orden.

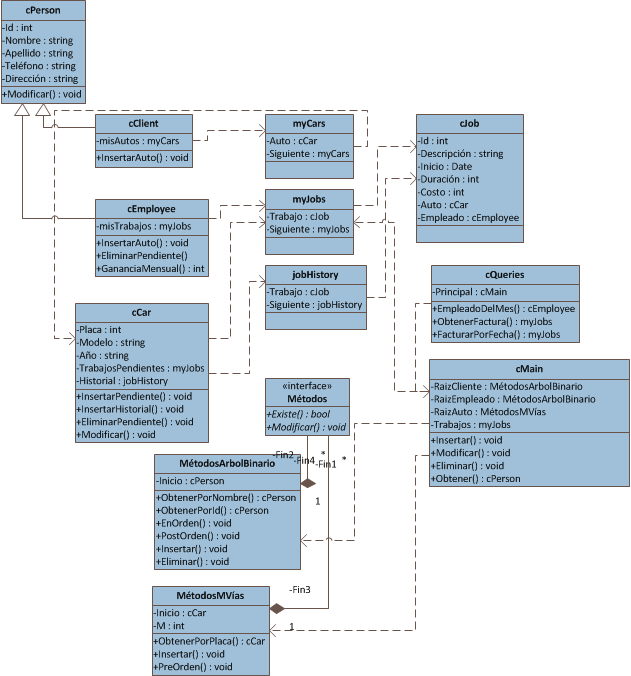
Los clientes deben tener al menos un auto asignado. Los trabajos que ya han finalizado deben mostrarse en una factura para que se puedan cancelar. Una vez cancelado un trabajo, los trabajos se deben mover al historial de cada auto y eliminarlos de los trabajos pendientes de los empleados.

El sistema debe permitir mostrar los trabajos que hayan finalizado para una fecha dada, mostrar el empleado del mes, o sea, el que haya ganado más en un mes y año dados. Se deben presentar los diez clientes más frecuentes del taller, tomando en cuenta el número de trabajos que tiene con cada auto. Finalmente, se deben mostrar los diez autos con más tiempo en el taller.

Se debe implementar la capacidad de leer información de un archivo con un formato pre-establecido. Este archivo debe tener información de clientes y sus autos respectivos.

Solución del Problema

**Diagrama de clases**



# Realización general de la aplicación

Para generar la aplicación, se realizó una construcción de clases y objetos a partir de las listas básicas: clientes, empleados, autos y trabajos. A continuación creamos las sub-listas de autos, trabajos e historial, las cuales agregamos a las anteriores en la estructura adecuada.

La idea general es aprovechar la capacidad que tienen las estructuras de árboles binarios y árboles de M vías al manejar información. Por esta razón las 3 clases principales del proyecto son estructuras de este tipo.

Para empezar a cumplir los requerimientos del sistema, creamos una serie de clases y sub-clases, además de una clase interface y una super-clase. Para ilustrar un poco más, presentamos la siguiente tabla con el nombre de cada clase y una breve descripción muy general de lo que contiene o realiza:

|  |  |
| --- | --- |
| Nombre | Descripción |
| cMCarNode | Nodo de M vías que contiene M – 1 autos y M ramas a nodos. |
| cBinaryTreeMethods | Métodos para los nodos binarios de la clase persona o sus sub-clases. |
| cCar | Clase auto, contiene la información del auto, junto con sus trabajos pendientes e historial (trabajos ya terminados y facturados). |
| cClient | Clase cliente, hereda las propiedades de la clase persona, además agrega una lista de autos. |
| cEmployee | Clase empleado, hereda a persona y posee, por su parte, una lista de trabajos pendientes. |
| cJob | Clase trabajo, posee la descripción del trabajo, así como el empleado al que corresponde y el auto afectado. |
| cMain | Clase principal, posee 4 listas principales y métodos polimórficos para insertar, modificar, eliminar y buscar. |
| cMWayTreeMethods | Métodos de árboles de M vías. En este caso, la clase nodo de M vías. |
| cMyCars | Mis autos, señala un auto. Permite crear listas independientes de autos. |
| cMyJobs | Mis trabajos, señala un trabajo. Útil para listar trabajos. |
| cPerson | Super-clase persona, permite almacenar información personal. |
| cQueries | Clase consultas, permite realizar otras operaciones distintas de los métodos tradicionales. |
| iMethods | Interface métodos. Distingue los métodos de modificar y existe. |

## Operaciones y consultas

A continuación, explicaremos los métodos necesarios realizados para el funcionamiento del programa. Para mayor facilidad se ilustran con diagramas de flujo, a excepción de los más sencillos, que consideramos no tiene mucha complejidad.

### Insertar clientes, empleados y autos

Para agregar un nuevo elemento en cualquiera de estas listas, es necesario determinar el orden en el que se desea insertar. Al ser clientes y empleados clases de personas y nodos binarios, sus métodos son los mismos, mientras que para los autos son distintos.

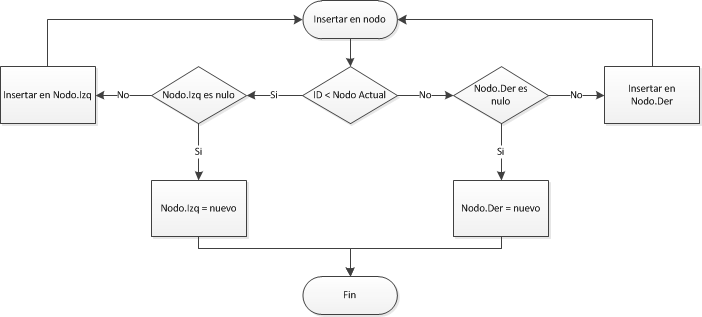


Diagrama Insertar en árbol binario

Cabe destacar que este método tiene dos pequeñas variaciones. En la primera condición que se encuentra, al trabajar con clientes, en vez de utilizar la cédula o ID, se pregunta por el nombre, ya que entre los requerimientos se pide que los clientes se encuentren ordenados por su nombre.

Sin embargo, a pesar de esa diferencia, ambas clases funcionan con esta misma estrategia, ya que son clases que heredan de la clase persona, la cual es un nodo binario (posee una rama izquierda y una derecha).

Luego, tenemos el método de inserción en M vías, para la clase Nodo de M vías la cual señala hacia los autos. Este método presenta una estructura similar a la presentada en el siguiente diagrama:

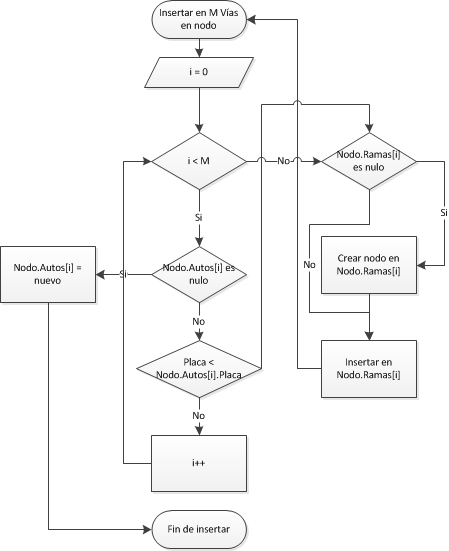


Diagrama Insertar en M vías

Este método es un tanto más complicado que el anterior, pero permite utilizar cada nodo para almacenar una cantidad mayor de autos de la que puede contener el árbol binario, así como más punteros de ramas del mismo tipo de nodo.

### Modificar clientes, empleados y autos

Los métodos de modificar son realmente simples, ya que para cualquier clase solo se debe buscar el objeto que posea el mismo identificador que se busca y cambiar los datos o información que posee. Adicionalmente, cada una de las clases en nuestro proyecto, posee un método de modificar el cual acepta un arreglo de nuevos valores que simplemente se toman en un orden predefinido y se utilizan en vez de los que ya tiene el elemento. Este método es polimórfico para todas las clases de árboles.

### Eliminar clientes y empleados

Estas operaciones son un tanto más complicadas de implementar que las de inserción, ya que se debe anticipar si se encuentra el nodo buscado en una de las ramas del nodo que se revisa, y luego se asigna uno de los hijos del eliminado a la posición que ocupaba el mismo.

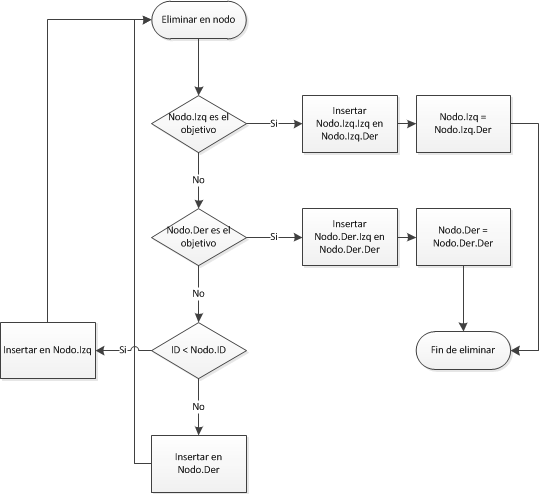


Diagrama Eliminar en árbol binario

### Asignar autos a clientes

Se crea un auto nuevo (ya que no puede haber autos sin cliente asignado) y se inserta en la lista de autos del cliente seleccionado. El método es muy simple y no necesita mayor explicación. Sin embargo, es necesario mencionar que, para lograr que no queden clientes sin autos, al insertar un cliente, debe crearse una lista de autos que no puede estar vacía. Esto se logra a nivel de interfaz.

### Asignar trabajos a los autos

Esta operación es similar a la anterior, ya que los autos poseen una lista de trabajos pendientes que posee un pequeño y sencillo método de inserción, al cual se le envía el nuevo trabajo. Al trabajar con el programa, se selecciona el número de placa del auto cuando se agrega un nuevo trabajo a un empleado.

### E:\Documents\ITCR\2010 - Semestre II\Algoritmos y estructuras de datos I\Proyectos\Segundo - Taller Automotriz 2\Diagramas\Insertar trabajo.pngAsignar trabajo a empleado

Diagrama Insertar trabajo

Para realizar esta operación se elige un empleado y se selecciona su lista de trabajos, ahí se puede insertar un nuevo trabajo, el cual se inserta automáticamente a la lista de trabajos pendientes del empleado, así como a la lista de trabajos del auto (como se mencionó en el método anterior).

### Facturar a una fecha por cliente

Para realizar esta comprobación de clientes y trabajos asignados a sus autos se sigue un proceso de búsqueda.

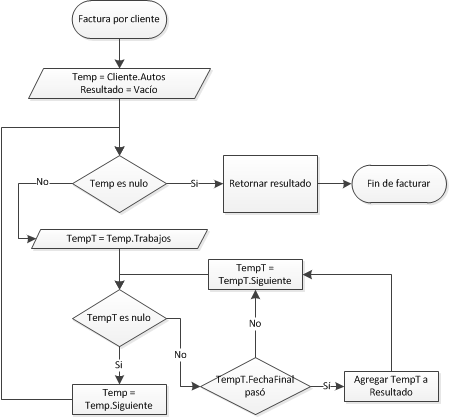
Para el cliente dado, se recorre su lista de autos y para cada uno de estos se selecciona su lista de trabajos pendientes. De cada trabajo se obtiene la fecha en la que finaliza y si esta fecha está delante de la seleccionada se agrega a una lista temporal de trabajos, la cual representará la factura en sí.

Diagrama Facturar por cliente

Este proceso se describe mediante el siguiente diagrama:

### Pagar una factura

Esta operación implica más que todo, mover y quitar elementos. Una factura es simplemente una lista de trabajos, así que para pagarla, se debe recorrer cada uno de sus elementos. Por cada elemento en la lista, se elimina de los trabajos pendientes del empleado, así como de los trabajos del auto asignado y se inserta en el historial del auto en cuestión. Adicionalmente, se agrega el costo del trabajo a las ganancias del taller.

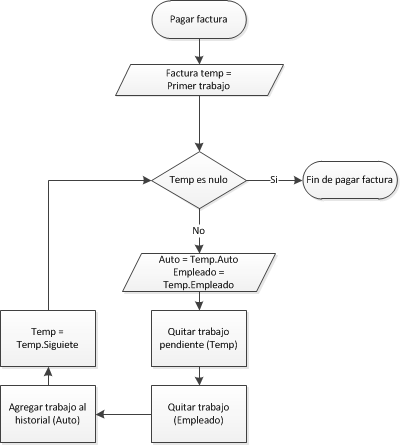


Diagrama Pagar una factura

### Leer de archivo

A la lista de clientes y de autos se pueden agregar elementos desde un archivo, siempre y cuando, este posea el formato adecuado. Este formato está estructurado por medio de algunos símbolos que permiten separar cada sección de los objetos almacenados en él.

Un punto (.) separa las propiedades de un objeto, ya sea cliente o auto. Una coma (,) separa cada cliente de sus autos y cada auto entre sí. Un punto y coma (;) separa cada cliente por aparte. El siguiente es un ejemplo de cómo debe presentarse un archivo para que pueda ser leído por el sistema.

Auto 2

Auto 1

Cliente

20683.alonso.vega.24588123.la tigra,2890.toyota.1999,2991.nissan.2004;

Para cada cliente se necesita proporcionar su cédula, nombre, apellidos, teléfono y dirección, en ese orden. Luego de la información personal, se puede agregar una coma e insertar los datos de un auto, utilizando su número de placa, marca y modelo. Utilizando nuevamente una coma, se pueden insertar más autos al mismo cliente. Para insertar otro cliente, se necesita un punto y coma y repetir el mismo procedimiento anterior.

El proyecto utiliza dos clases para leer e insertar desde un archivo: un lector de archivo que obtiene el texto de un archivo dado y un interpretador, que recibe el texto y lo decodifica de la forma mencionada anteriormente, luego toma cada parte (cliente o auto) y lo inserta en la clase principal.

Para este método no presentamos un diagrama debido a la complejidad que presenta mostrar cada comprobación y cada ciclo que se realiza. Además la tarea de leer de un archivo se basa en 4 métodos distintos (leer el texto de archivo, recorrer el texto y buscar los puntos y comas, luego, entre comas interpretar la primera sección como cliente y el resto como autos) y se necesitan revisar varios símbolos.

### Mostrar trabajos pendientes por día

Muy similar a facturar, ya que simplemente se recorre la lista principal de trabajos y se buscan los que hayan vencido para la fecha dada. A continuación, mostramos gráficamente los procedimientos realizados en este método:

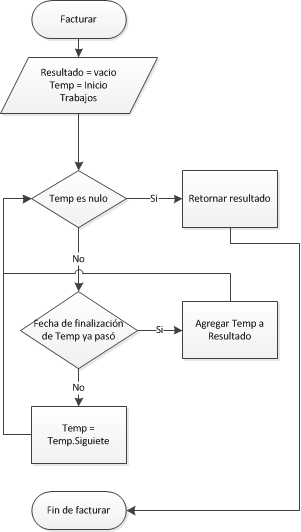


Diagrama Trabajos por fecha

### Listar autos del cliente, trabajos pendientes e historial del auto

Este es otro método simple en el cual se lee la lista de autos del cliente que haya sido seleccionado. Se muestra la información de cada auto. Para ver la lista de trabajos de un auto, se selecciona por número de placa y desde ahí se pueden ver sus trabajos pendientes así como el historial.

### Empleado del mes

Con cada empleado se puede determinar las ganancias que ha tenido buscando entre los trabajos los que hayan sido realizados por él y que tengan como fecha de finalización el mes y año que se le dan. Para ver cuál es el que ha ganado más, solo se debe comparar cada uno y seleccionar el que haya obtenido la mayor ganancia.

Esa comparación la realizamos de la forma siguiente:

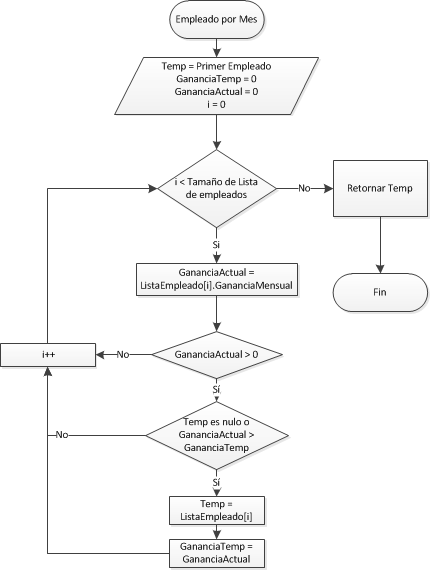


Diagrama Empleado del mes

## Interfaz

La presentación del programa es sencilla y básica. Se centra en las 3 listas principales y cómo se puede trabajar con ellas. Adicionalmente, se puede acceder a la ventana de consultas, la cual contiene los controles para revisar algunos de los métodos requeridos. También se puede utilizar la ventana principal para insertar clientes y autos desde un archivo.

Para la mayor parte de la información que se debe mostrar, se utilizan *grids* o tablas, las cuales permiten mostrar una buena cantidad de información de manera ordenada. Además se utilizan diálogos de advertencia o de error cuando algún campo no tiene el formato adecuado, para seleccionar el archivo cuando se quiere leer información de él y para las ventanas de listas.

Análisis de Resultados

# Operaciones

|  |  |
| --- | --- |
| Operación | Estado |
| *Insertar Autos* | *Completado* |
| *Insertar Clientes* | *Completado* |
| *Insertar Empleados* | *Completado* |
| *Modificar Clientes* | *Completado* |
| *Modificar Empleados* | *Completado* |
| *Modificar Autos* | *Completado* |
| *Borrar Empleados* | *Completado* |
| *Borrar Clientes* | *Completado* |
| *Mostrar cada lista en el orden requerido* | *Completado* |
| *Asignar trabajo a auto* | *Completado* |
| *Asignar trabajo a empleado* | *Completado \** |
| *Asignar autos a clientes* | *Completado* |
| *Facturación* | *Completado* |

*Análisis de Resultados funciones base del sistema.*

*\* No se evalúa si el empleado tiene un trabajo asignado en la fecha dada.*

# Consultas

|  |  |
| --- | --- |
| *Consulta* | *Estado* |
| *Mostrar Trabajos pendientes de un auto* | *Completado* |
| *Generar y mostrar factura de un cliente* | *Completado* |
| *Mostrar los trabajos pendientes dada una fecha* | *Completado* |
| *Listar autos en el taller, y los trabajos de éstos* | *Completado* |
| *Listar todo el historial de trabajos en un auto* | *Completado* |
| *Listar todos los trabajos asignados a un empleado* | *Completado* |
| *Mostrar empleado del mes* | *Completado* |
| *Listar los diez autos que han permanecido más tiempo de mayor a menor en el taller* | *No Completado* |
| *Listar los diez clientes más frecuentes* | *No Completado* |

*Análisis de Resultados de consultas.*

*Algunas consultas no se completaron debido a la falta de tiempo para finalizar el proyecto, debido a que necesitaban ciertas modificaciones en las demás clases para poder lograr implementarlas.*

Conclusiones y Recomendaciones

* Para este proyecto se evalúan las clases de árboles binarios, así que consideramos que quizá el proyecto no requiere gran utilidad de estas clases, sino que sólo se necesitan para manejar listas. Fue más necesario conocer el manejo de listas simples que el de árboles binarios. Quizá algún sistema de búsqueda mostraría la mayor eficiencia que tiene un árbol sobre una lista simple.
* El lenguaje Java es muy útil para diseñar cualquier software, ya que permite ejecutarse en cualquier sistema operativo que soporte la máquina virtual de Java, sin embargo, se nota el retraso en la ejecución debido a la forma tan indirecta con la que trabaja.

Literatura citada

1. The Java Tutorials. *How to Use Tables*. Consultado en noviembre, 16, 2010 en <http://download.oracle.com/javase/tutorial/uiswing/components/table.html>
2. The Java Tutorials. *How to Make Dialogs*. Consultado en septiembre, 16, 2010 en <http://download.oracle.com/javase/tutorial/uiswing/components/dialog.html>
3. The Java Tutorials. *Interfaces and Inheritance*. Consultado en noviembre, 10, 2010 en <http://download.oracle.com/javase/tutorial/java/IandI/index.html>