DESAFÍO 2

Juan Luis Becerra Díaz

Prof. Aníbal Guerra

Informática II

Universidad de Antioquia

2024

1. Descripción del problema.

La empresa TerMax es líder en la distribución de combustibles en Colombia, operando una extensa red de estaciones de servicio a nivel nacional. Cada estación almacena y vende tres tipos de combustibles: Regular, Premium y EcoExtra. El combustible se almacena en un único tanque y la comercialización se hace mediante maquinas surtidoras. El objetivo de este informe es analizar y proponer una solución que permita gestionar eficientemente la red nacional, las estaciones de servicio, los surtidores y las transacciones, además de verificar fugas y simular ventas en la red.

2. Consideraciones iniciales

La esencia del proyecto yace en el establecimiento y relación de las clases, es decir, de qué manera podemos conectar y manipular las agrupaciones de objetos. Sabemos que las estaciones de servicio tienen códigos identificadores, al igual que las máquinas surtidoras, si las máquinas surtidoras pertenecen a las estaciones cuál será la mejor forma de correlacionar estaciones con surtidoras, de conectar el tanque con las surtidoras, o de qué manera agrupamos las estaciones en una red nacional. Otro aspecto para tener en cuenta es el manejo de las transacciones, aunque se delimitara bien el manejo de datos durante la ejecución del programa estos datos quedarán eliminados al finalizar el programa, lo que no es una opción si estamos trabajando en un ámbito real, y así como con las transacciones, la información de las estaciones, las surtidoras y el tanque me parece pertinente que quede almacenada aún después de finalizar el programa. Por último, el manejo de las estructuras de datos que almacenarán los objetos, por ejemplo, al momento de agregar o eliminar estaciones o surtidoras, tendremos que redimensionar o usar otra estrategia.

En resumen, considero tres ideas importantes para tener en cuenta: la implementación de las relaciones entre las clases, el manejo de archivos de texto para almacenar información y el manejo de las estructuras de datos para agregar o eliminar.

3. Análisis del problema

Clases consideradas:

Clase transacción: si bien esta clase, a parte de los métodos *set* y *get* no parece tener otros métodos, tener objetos que guarden la información de las transacciones nos facilitará la manipulación de las estas.

Clase tanque: instancias de tipo tanque, permite dos cosas: que la implementación se asemeje a la realidad, y segundo, facilitar la relación entre surtidoras y estaciones y el manejo de las cantidades de cada combustible.

Clase surtidora: clase importante con varios atributos y funcionalidades especificadas en la guía.

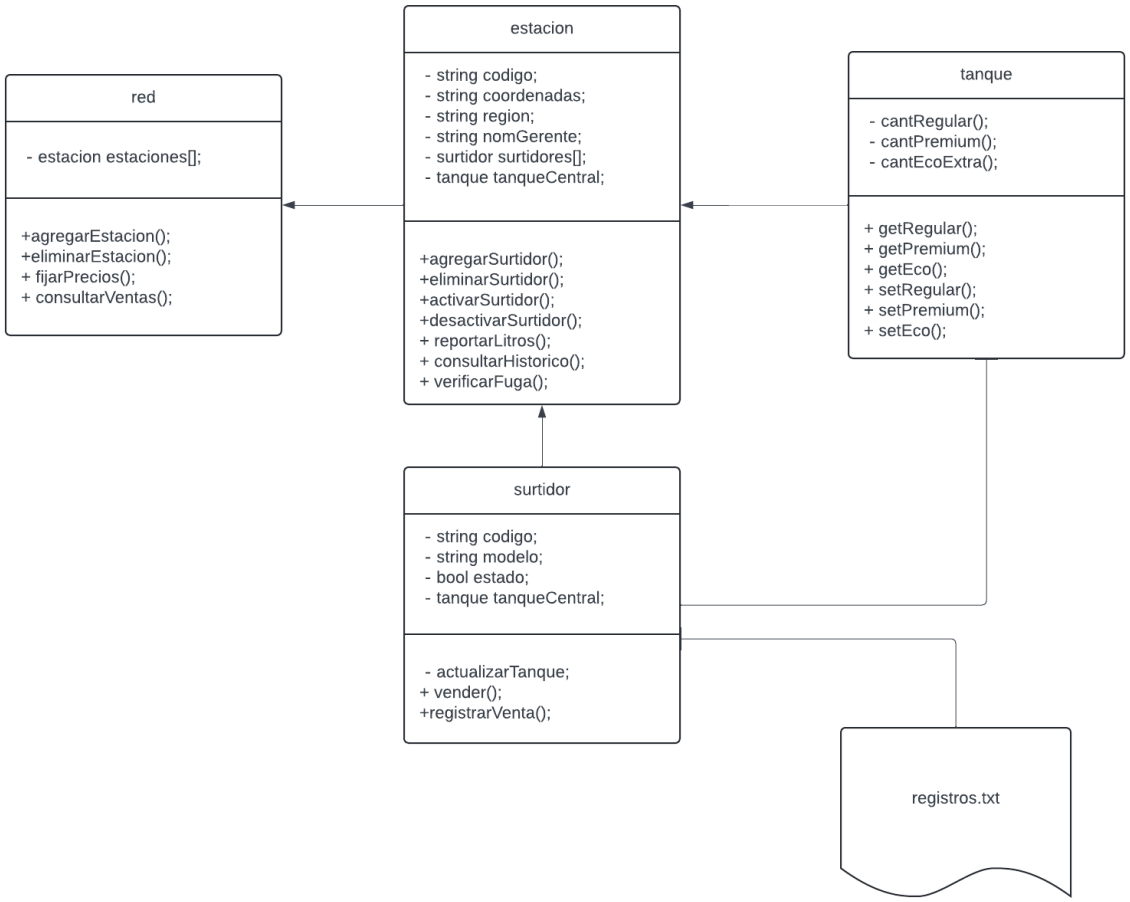
Clase estación: guardará el arreglo de surtidores y un tanque.

Clase red: guardará el arreglo de estaciones.

Manejo de archivos:

El manejo de archivos se considera para guardan la información de las transacciones, además de la información de las estaciones y las surtidoras.

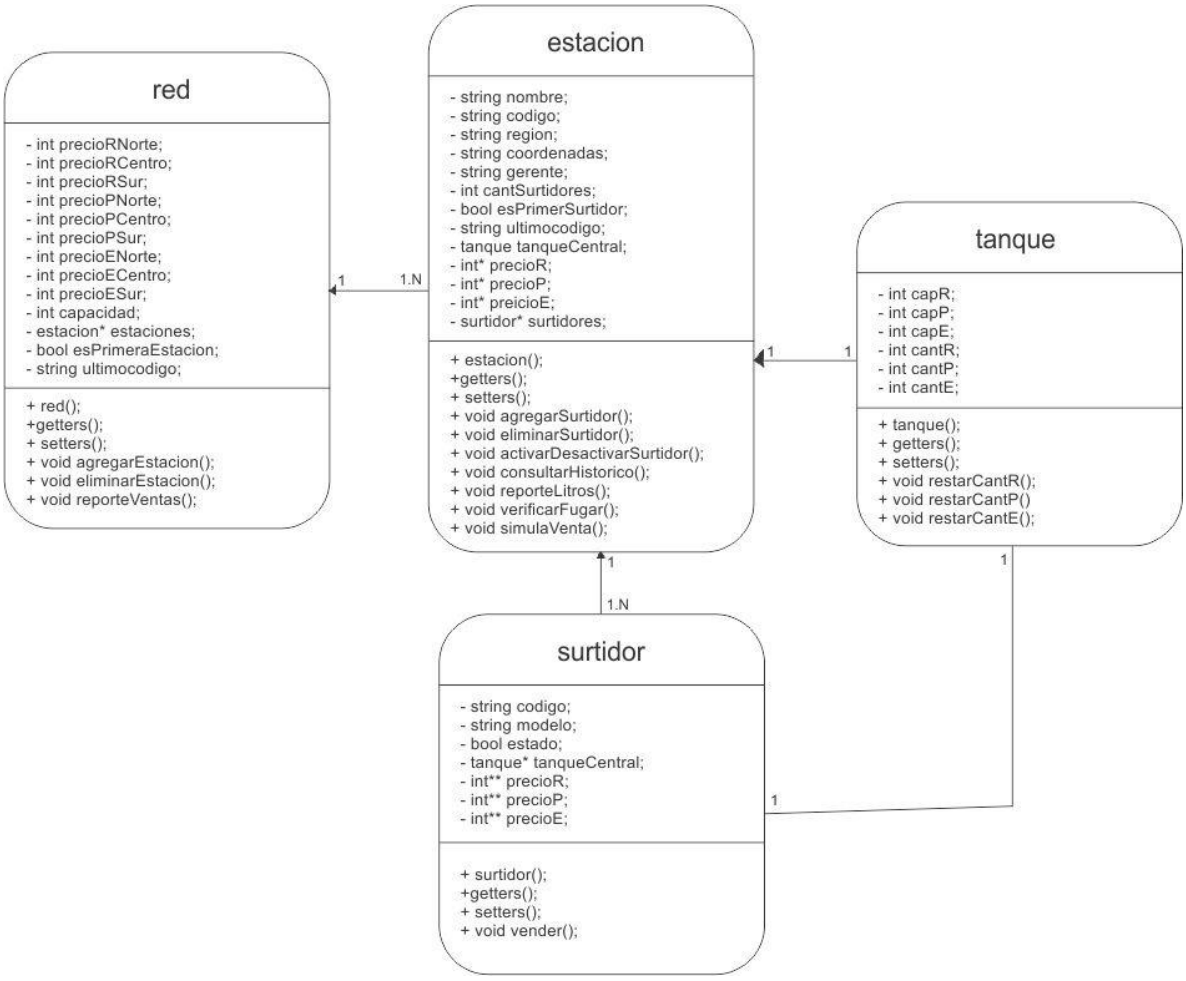
4.



5. Evolución del análisis y la solución

- Se había considerado una clase transacción para registrar las ventas, sin embargo, teniendo en cuenta que las transacciones se guardarían en un archivo de texto, veo innecesario almacenar los datos en un objeto antes mejor guardarlos directamente en el archivo de texto, además no tendremos que redimensionar el arreglo de transacciones, cada vez que se realice una transacción lo que supone un ahorro significativo de recursos de procesamiento. Y si al registro se le añade el código de la maquina surtidora no habrá inconvenientes al recorrer el archivo de texto y discriminar por surtidoras o estaciones para calcular las ventas totales. Para este caso consideramos un código general, de la forma 000101 – los primeros 4 dígitos corresponderán a la estación los últimos dos a la surtidora, entonces 000101 será la primera surtidora de la primera estación, 000512 será la doceava surtidora de la quinta estación, y así.

6.



`agregarEstacion();`

Pide los datos al usuario y redimensiona el arreglo de estaciones en uno más.

`eliminarEstacion();`

Recibe el código de la estación a eliminar y copia redimensiona el arreglo en uno menos sin copiar la estación que coincida con el código.

`reporteVentas();`

Lee la información del archivo de registros y acumula el numero de ventas discriminando por estación y por tipo de combustible.

`agregarSurtidor();`

Pide los datos al usuario y redimensiona el arreglo de estaciones en uno más.

`eliminarSurtidor();`

Recibe el código del surtido a eliminar y copia redimensiona el arreglo en uno menos sin copiar el surtidor que coincida con el código.

`activarDesactivarSurtidor();`

Recibe el código del surtidor y cambia su estado.

`consultarHistorico();`

Lee el archivo donde se guargan los registros y discrimina la información de las transacciones por surtidor e imprime por pantalla.

`reporteLitros();`

Lee la información del archivo de registros y acumula los litros vendidos discriminado por tipo de combustible.

`verificarFugas();`

Lee la información del archivo de registros y acumula la cantidad de litros vendidos, la suma con la cantidad del tanque central y compara con la capacidad inicial.

`simularVenta();`

Escoge un surtidor aleatorio.

`vender();`

Asigna valores aleatorios a todos los valores de la transacción, guarda en el archivo de registros e imprime por pantalla la información de la transacción.

`restarCantR();`

Recibe un entero que resta a la cantidad de combustible regular del tanque.

`restarCantP();`

Recibe un entero que resta a la cantidad de combustible premium del tanque.

`restarCantE();`

Recibe un entero que resta a la cantidad de combustible eco extra del tanque.

7. Problemas en el desarrollo

- Presento problemas con los constructores por defecto de las clases red, estaciones y surtidores, al ser clases anidadas una a la otra, el uso de punteros para la clase surtidor y la clase estación no encuentro donde apuntarlos si aún no existe el valor donde deberían apuntar.