Informe Desafio #2.

Alejandro Stev Ocampo R

Juan David Jimenez M

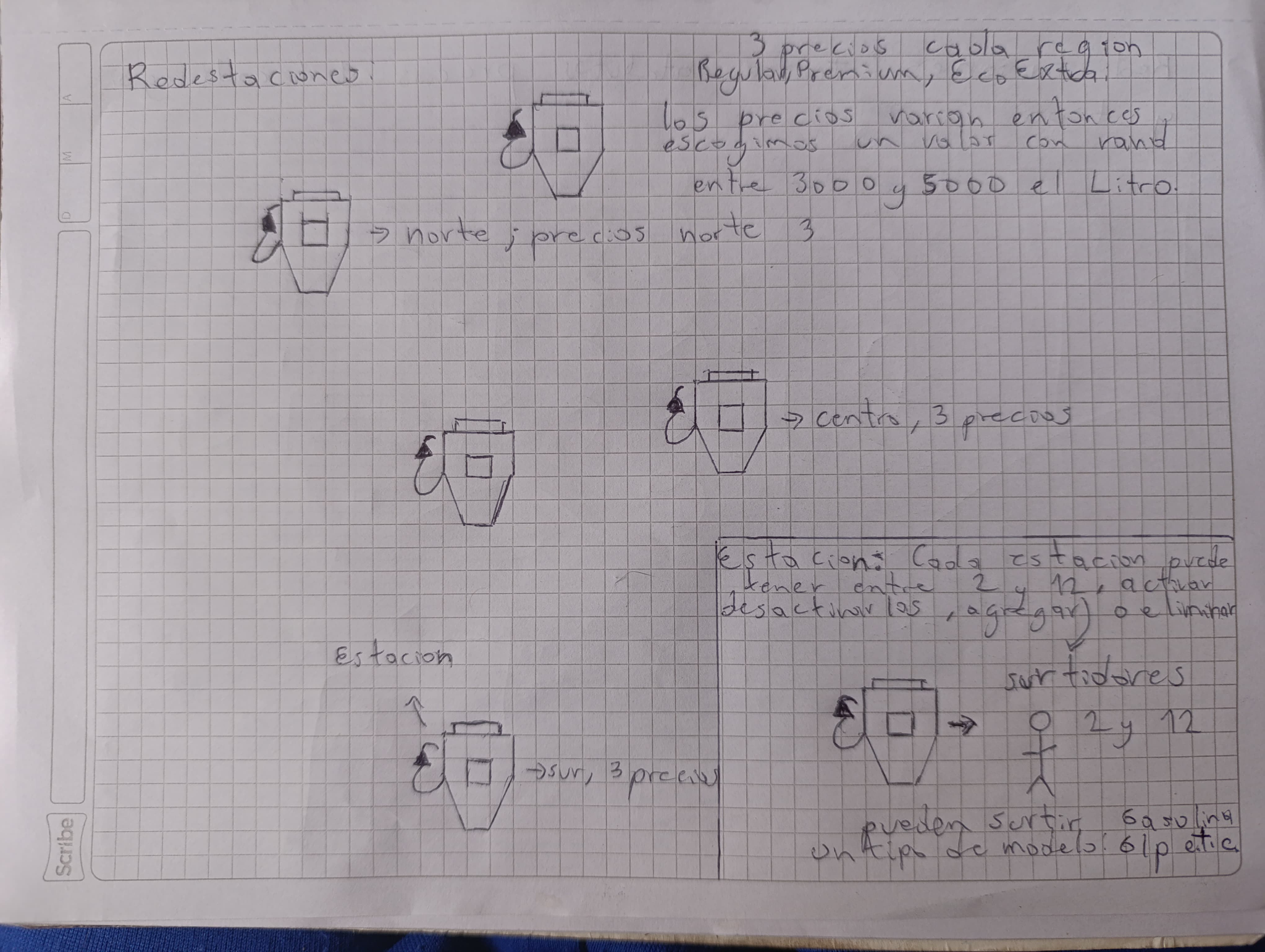
a. Análisis del problema y consideraciones para la alternativa de solución propuesta.

Para empezar iremos escribiendo poco a poco como analizamos el programa viendo qué consideraciones debemos tener en cuenta y cuales pueden guiarse por una mejor alternativa, así progresaremos poco a poco en la evolución del problema planteado.

Utilizaremos los herramientas vistas en clase haciendo uso de ellas lo mejor posible, de las cuales están el uso de clases, sobrecargas, punteros , memoria dinámica para la eficiencia, el diagrama de clases para un mayor orden, utilizando los pilares de la POO como la abstracción,encapsulamiento,polimorfismo entre otras cosas como las relaciones de clase a clase que se incluyen en el diagrama de clases.

Como primer análisis crearemos clases, una para la gestión de la red ,para agregar estaciones eliminar entre otras cosas que piden,la segunda sería las estaciones de servicio (E/S) o bombas,por tercero sería las surtidoras que por supuesto teniendo cuidado en cosas como solo se puede crear si se puede adscribir a una estación existente, o de igual manera si una surtidora está asociada a una (E/S) no se podra eliminar.

Todo esto se logrará en un trabajo en equipo implementado el uso de getters para el retorno de un valor a consultar ,setters en caso de querer modificar alguna variables, en caso de que si se pueda hacer ,constructores para la inicialización de variables (Hacerlo de forma manual para evitar problemas y más por ejemplo cuando se habla de memoria dinámica), y los destructores que se encargaran de la liberación de memoria para tener un mejor uso de la memoria y por lo tanto una adecuada eficiencia.Además el uso de otras funcionalidades especialmente para los numerales lll y lV de verificación de fugas y la simulación de ventas respectivamente.



La clase red estaciones tendrá como atributos Los precios de cada región(norte,centro y sur), y de cada tipo ,(regular,premium y eco extra), estos se inicializan con un precio aleatorio con la función de rand en valores entre 3000 y 5000 el Litro permitiendo que se puedan fijar de diferentes maneras a que el precio varia por dias, tmb tendrá un arreglo dinámico de objetos Estación para permitir la creación de estaciones o eliminarlas según quiera el usuario,con el uso de los constructores getters setters y liberando la memoria etc, luego tendrá un arreglo estático de los tipo de combustible (regular,premium y eco extra),.y el número de estaciones que cambiara segun se creen o eliminen estaciones

La clase estación tendrá como atributos el nombre, el código identificador, la región ,coordenadas (algo simples en un plano y ) el número de surtidores para cuando se agreguen eliminen surtidores,también un arreglo animación de objetos surtidor, que al igual de la estaciones para permitir la creación de surtidores o eliminarlas según quiera el usuario,con el uso de los constructores getters setters y liberando la memoria etc ,luego tendrá la capacidad que pueden almacenar los tres tanques entre 100 y 200 un valor aleatorio con la función rand y otro capacidad la capacidad tanque inicial que nos servirá para el sistema de verificación de fugas.

La clase surtidor tendrá dos atributos , un código identificador y el modelo de máquina por ej: Gasolina Glp, etc.También un arreglo de 3 que contiene los litros vendidos de cada tipo de combustible. Además cada clase tendrá sus métodos correspondientes para el manejo de nuestros atributos según lo pedido. Para la parte de simular venta de un surtidor hemos decidido utilizar un archivo de txt que guardará las transacciones que se hagan de un surtidor asociado a una estación.

Para varios puntos se requiere escoger una estación el programa la encontrará con su código identificador,que el usuario ingresará dependiendo que estación quiera utilizar.

Métodos abajo.en el ©.

b. Diagrama de clases de la solución planteada

//Vamos a poner aqui diagrama o en Git

c. Algoritmos implementados debidamente intra-documentados.

(Nota no todos los métodos estan aca, mas que todo los mas importantes).

Incluyendo los getters setters, constructores y destructor de las clases tendremos tmb ademas métodos para la realización de nuestro programa

Clase surtidor:agregarLitrosVendidos este método permite sumar los litros vendidos al total del tipo de combustible

Clase estación:Nos gustaría especificar aqui un getsurtidor; hay dos una sobrecarga cuando este recibe un parámetro se usa en la parte de simular venta para que ese surtidor específico sea el de la venta,

crearMaquinas(), Crear el arreglo dinámico de los surtidores. cuando se activa los surtidores se usa este método y se crea con un id y un modelo del surtidor.

agregar\_eliminar\_surtidor( bool, nuevoSurtidor,indiceEliminar), se pasan tres parámetros un bool tru o false si se quiere agregar o eliminar y el surtidor a agregar y si se quiere eliminar un índice a eliminar,

asignarTanques(); asigna la capacidad entre 100 y 200 de cada tanque son (3) de una estación.

restarLitros(tipodecombustible,litrosVendidos),restar los litros vendidos a un tanque específico,

verificarFugas(), toma la cantidad disponible de cada tanque y los litros vendidos de cada tanque,

y esto debe ser mayor a l 95% de lacapacidad inicial del tanque.

Clase redestaciones:agregarestacion(),recibe los atributos de una estación y agrega la estación en el arreglo de la estaciones..

generarNumeroAleatorio();para generar un número aleatorio para los precios que varían (los precios entre 3000 y 5000 el Litro).

eliminarEstacion(), toma el codigo de identidentifiacion y verifique si hay surtidores , si no hay elimina la estacion.

d. Problemas de desarrollo que afrontó

Pues como en todas partes siempre hay problemas he aquí unos, No sabíamos muy bien al principio como empezar ya que no entendiamos muy la sintaxis de incluir un objeto como atributo de otra clase como el ej de que la red tenía como atributo unas estaciones y la estación como atributo unos surtidores,

Tambien era algo dificil el hecho de hacer el codigo tratando de entender lo que había hecho nuestro compañero , y hacia mas lento el proceso , por los que nos tocó buscar algo en lo que pudiéramos trabajar a la vez y así avanzar mejor para ir comunicándonos con nuestro compañero para saber por que camino irnos o preguntarle algo que no se supiera. La función simular venta fue la que mas nos costo ya que esta tenia metodos de varias clases y pensábamos hacerlo en un método de la clase estación pero necesitaba los precios de la clase estación, entonces nos decantamos por hacer otro archivo .cpp y crear la función que invocara todos los métodos necesarios

e. Evolución de la solución y consideraciones para tener en cuenta en la implementación.

Empezamos con la parte fácil yendo de los más pequeño a lo más grande empezamos con la clase surtidor que es la más simple haciendo los getters respectivos setters y constructores y destructores

Luego lo mismo para las otras clases, lo que sabíamos que teníamos que ir implementando, así luego poco a poco íbamos avanzando , preguntando a otros equipos y una vez que empiezas y tienes las bases ya entiendes mejor podemos hacer la parte más difícil, Nos pareció que aunque este desafío estuvo un poco más largo , en verdad era bastante más fácil que el anterior ya que si seguimos bien lo que nos dice la guia nos va mejor, Utilizamos variables escogidas de acuerdo a lo que se necesitaba por ej si no era necesario un int mejor hacíamos un short int etc, también el uso de memoria dinámica y punteros a referencias para tratar de hacerlo los mas eficiente y evitar copias innecesarias, de esta manera pudimos desarrollar el desafio.