**DESAFIO 2.**

Para satisfacer los requerimientos de la empresa TerMax y considerando el paradigma de programación orientada a objetos se considera como primer paso determinar a partir del enunciado cuales clases se van a necesitar para modelar el sistema. Tras el análisis inicial del problema se consideran en primer lugar 4 clases, la clase red nacional, la clase estación de servicio, la clase surtidor y la clase tanque central (estas clases se pueden ver sujetas a cambios durante el paso de los dias), en base a esto surge una pregunta fundamental, ¿Qué son las islas?, algo clave a tener en cuenta es el contexto del problema planteado y la delimitación de la realidad, es decir, con las funcionalidades que se necesitan ¿es verdaderamente esencial tener a las islas como clases?

Tras meditar sobre ello se concluye que no es necesario en este contexto, tal vez en un problema con requerimientos distintos podría ser una estrategia viable y fundamental para el funcionamiento del sistema, sin embargo el programa que se nos solicita se basa en la gestión de las estaciones de servicio, los surtidores y las transacciones realizadas, haciendo que modelar las islas como clases sea algo innecesario en el problema planteado, las islas se definirán mediante funciones/métodos que se determinaran mas tarde en el desarrollo.

Tras haber definido esto, se analiza que relación existe entre las clases mencionadas, en la clase de red nacional como se especifica en el documento solo se va a crear una instancia, por ello la red nacional va a ser el “todo”, la red nacional puede tener muchas estaciones de servicio, es decir, las estaciones de servicio hacen parte de la red nacional, mismo caso que los surtidores con las propias estaciones de servicio, pueden haber muchos surtidores por estación y los surtidores hacen parte de las estaciones, finalmente los tanques centrales son una relación uno a uno con las estaciones, por cada estación hay un único tanque central.

La instancia de la red nacional se va a crear con unos precios para cada tipo de combustible predeterminados, si estos no son cambiados por el usuario se va a operar en base a ellos. El usuario va a tener la opción de cambiar estos precios cuando quiera, hay que tener en cuenta que según el enunciado del desafío los precios varían según la región de la estación de servicio, este ajuste de precios se realizara de forma automática con un ajuste porcentual.

El ajuste se determina con un acercamiento a lo que pasa en la realidad, en la región norte del país los precios de los combustibles tienden a ser mas altos por los costos de transporte desde el interior, la infraestructura y la logística requerida, por tanto, en el diseño del programa se aumentara un 5% al precio de todas las categorías de combustible si la estación de servicio se encuentra en la región del norte. Si la región es centro se conservarán los precios establecidos para la red nacional ya que en el interior del país se encuentran las principales refinerías, reduciendo los costos de distribución. Finalmente, si la región es sur se reducirán los costos de todos los tipos de combustible un 3%, esto debido a que es una región con menos demanda y que cuenta con subsidios del estado en compensación de las dificultades económicas del área y la lejanía.

Para crear y eliminar estaciones de servicio se va a hacer uso de un archivo txt donde se guardará la información de las instancias creadas, todas las estaciones van a cumplir un formato predefinido en el archivo, para así luego poder ser manipuladas. Las estaciones deben tener nombres distintos y coordenadas distintas para poder ser creadas, se identificarán por un ID que se conseguirá mediante un atributo estático, este se va a actualizar siempre que se cree un nuevo objeto, asegurando así distintos identificadores siempre.

Para el almacenamiento predeterminado del tanque central (que debe ser un numero entre 100 y 200 para cada categoría) se hace uso de la librería random, con ella vamos a poder generar un numero en este rango de manera totalmente aleatoria, haciendo que cada estación creada tenga capacidades distintas siempre, esto no quita que el usuario pueda asignar estas capacidades de almacenamiento cuando quiera manualmente.

**Desarrollo.**

Durante el desarrollo se realizan dos cambios en la estructura y lógica del programa con respecto al análisis inicial, el primero es la eliminación de la clase tanque central; tras mirar más a profundidad los requerimientos y los posibles atributos de las clases nos percatamos de que el tanque central era mas optimo asignarlo como un atributo de la clase estación de servicio, esto debido a que en el enunciado no se especifican muchas posibles tareas que pueda resolver una clase tanque central. Como cada estación de servicio posee un único tanque central podemos ver los almacenamientos individuales de combustible como atributos, pudiendo reasignarlos a través de setters simplemente.

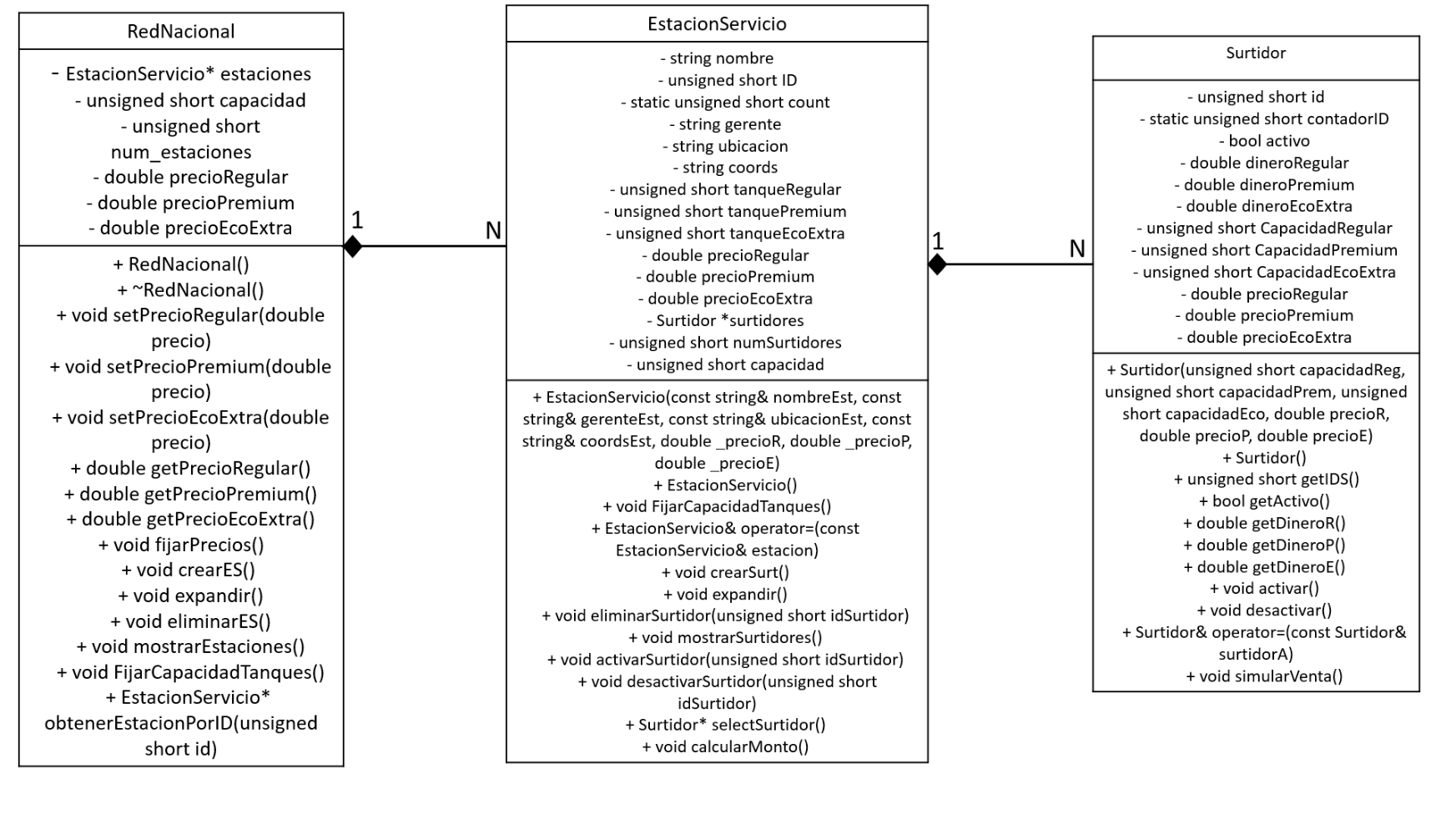
El otro cambio es el descarte de la idea de operar la información a través de archivos.txt, se determinó que la forma adecuada y optima de hacerlo era a través de arreglos dinámicos. Los archivos presentan la ventaja para almacenar la información, sin embargo, no se deben usar como mesa de trabajo, en este caso, debíamos no solo guardar datos sino modificarlos también, por tanto, guardarlos en arreglos es una decisión mas indicada. Los arreglos mencionados son atributos de 2 clases, RedNacional y EstacionServicio. Podemos imaginar la solución del problema como las muñecas rusas matrioshka; la red nacional es la muñeca más grande, no es contenida por ninguna otra clase (por ser solo una instancia), la clase estación de servicio esta encapsulada en la clase red nacional, es decir, la red tiene una colección de estaciones de servicio (el arreglo dinámico como atributo), para dar final a la analogía los surtidores están encapsulados en las estaciones de servicio, siendo la parte mas pequeña de esta encapsulación, por tanto las estaciones de servicio tiene un arreglo dinámico de surtidores.

El tamaño de estos arreglos va a depender exclusivamente del cliente, cada vez que la reserva de memoria sea insuficiente para la cantidad de datos, esta se va a actualizar automáticamente. El arreglo de estaciones no tiene una capacidad máxima definida, esto debido a que en el enunciado del problema nunca se especifica hasta cuantas estaciones puede haber, por el contrario, el arreglo de surtidores tiene un límite de 12 elementos.

El main se compone del uso de los métodos definidos en el resto de archivos para así llevar a cabo las tareas pedidas, a continuación, se explicarán los algoritmos fundamentales que fueron implementados en la solución:

* Creación de estaciones: La creación de estas instancias ocurren de manera similar, teniendo algunas diferencias; las estaciones requieren de algunos parámetros para su creación a través del constructor, estos datos deben ser brindado por el usuario (nombre de la estación, nombre del gerente, región y coordenadas asociadas), al ser todos unos datos tipo string se pasan como parámetro por referencia, ahorrando el proceso de crear una copia para cada uno. Si los datos introducidos por el usuario son validos en el contexto del problema se creará dicha estación de servicio y se añadirá automáticamente al arreglo dinámico de la red nacional, donde luego va a poder ser consultado para otros procesos o inclusive para ser eliminado.
* Creación de surtidores: El caso de los surtidores es muy parecido, lo único que cambia es que su creación con el constructor no requiere de tantos parámetros dados por el usuario, siendo una creación más rápida y automática. Para su creación solamente se requiere que el usuario digite el código de una estación de servicio existente, a esta se le asignara el surtidor deseado con un ID y actividad por defecto, este puede ser desactivado cuando el usuario lo desee, este proceso es a través de un setter ya que la actividad de un surtidor es un atributo. Después de la creación se mete en el arreglo dinámico de la estación de servicio dada.
* Hay dos algoritmos sencillos que operan de maneras parecidas en diferentes áreas, uno de ellos se encarga de asignar los precios a los tres tipos de combustible según lo que el usuario desee (siempre y cuando sea mayor que 0) y el otro se encarga de lo mismo, pero con la capacidad de cada tanque, esta capacidad debe estar entre 100 y 200 litros. Ambos procesos se realizan con algoritmos que recurren a los setters.
* La simulación de ventas es un método de los surtidores ya que estos tienen acceso a la capacidad y al precio del combustible. Como se mencionó en el análisis inicial depende de la zona donde se haya creado la estación de servicio es el precio que tendrá el combustible, el algoritmo lo que hace es multiplicar el precio relativo a la región por la cantidad de combustible, acumulando la cantidad de dinero ganada según el tipo de combustible para luego poder ser mostrada en la opción de ver las ventas por estación de servicio.
* Los destructores mas importantes son los de la clase RedNacional y EstacionServicio, esto debido a que son clases que usan la memoria dinámica, por tanto, en los destructores debe estar la liberación de la misma para evitar que la memoria se fragmente.

Los problemas que se tuvieron en el desarrollo fueron referentes a las ventas, por nuestro modelo de solución se nos dificultó encontrar la manera de guardar cada transacción por surtidor para luego poder ser visualizada, la posible idea de solución que se planteó fue el uso de archivos ya que en este caso sería muy útil para simplemente almacenar toda esa información y luego poder ser impresa por pantalla. La otra principal dificultad fue la detección de fugas ya que no encontramos el modelo ni la forma de incorporar esta funcionalidad al código fuente realizado.

**Diagrama de clases:** En el siguiente diagrama se pueden visualizar las relaciones entre las clases, su cardinalidas, atributos y métodos.