DESAFIO NÚMERO 2

**Nombre del estudiante:** Edwin Piedrahita Piedrahita.

**Numero de documento:** 1042764630.

**Nombre del docente:** Augusto Salazar Jiménez.

**1. Análisis del problema:**

El problema consiste en desarrollar un sistema de gestión para una red de estaciones de servicio de combustible utilizando Programación Orientada a Objetos (POO). La empresa TerMax necesita administrar eficientemente sus estaciones de servicio, surtidores y transacciones de venta de combustible a nivel nacional en Colombia.

**Elementos principales:**

* Red nacional de estaciones de servicio
* Estaciones de servicio individuales
* Surtidores de combustible
* Tanques de almacenamiento
* Transacciones de venta

**2. Consideraciones para la solución:**

**a) Estructura de clases:**

* Clase para la red nacional de estaciones
* Clase para estaciones de servicio individuales
* Clase para surtidores
* Clase para tanques de almacenamiento
* Clase para transacciones

**b) Funcionalidades requeridas:**

* Gestión de la red (agregar/eliminar estaciones, calcular ventas, fijar precios)
* Gestión de estaciones (agregar/eliminar surtidores, activar/desactivar surtidores, consultar transacciones)
* Sistema de verificación de fugas
* Simulación de ventas

**c) Manejo de datos:**

* Usar estructuras de datos eficientes para almacenar y acceder a la información de estaciones, surtidores y transacciones
* Implementar un sistema de generación y gestión de códigos identificadores

**d) Interfaz de usuario:**

* Desarrollar un menú interactivo en la terminal para acceder a las diferentes funcionalidades

**e) Lógica de negocio:**

* Implementar restricciones lógicas (no eliminar estaciones con surtidores activos)
* Manejar diferentes categorías de combustible y sus capacidades
* Gestionar precios por región
* Simular ventas con asignación aleatoria de surtidores y cantidades

**f) Consideraciones de diseño:**

* Utilizar principios de POO como encapsulamiento, relaciones, funciones amigas, sobrecarga y uso de plantillas
* Asegurar la modularidad y reutilización del código
* Implementar manejo de errores y validaciones

**g) Rendimiento:**

* Optimizar el acceso a datos y operaciones frecuentes
* Considerar la escalabilidad para manejar un gran número de estaciones y transacciones

**h) Pruebas:**

* Desarrollar casos de prueba para validar cada funcionalidad
* Incluir pruebas para escenarios límite (tanques vacíos, capacidad máxima)

**i) Documentación:**

* Crear un diagrama de clases UML
* Documentar adecuadamente el código y las funcionalidades