Desafio II

Informática II

Eder Luis Polo Humanez

Isabella Carvajal Cuartas

Universidad de Antioquía

Semestre 2024-2

Según nuestro análisis para el desarrollo de este problema consideramos necesario la división de las tareas en momentos, en este caso, dividiéndolas en 2 para una mejor organización y mejor entendimiento del programa.

Primer momento.

1. En este apartado realizaremos la creación de las clases que consideramos necesarias, junto con demás funciones que estas clases necesiten.
   1. Creación clase ventas: Esta clase tendrá los siguientes atributos:
2. Fecha (la cual usaremos una librería para poner la fecha actual).
3. Hora (Usaremos librería para calcular hora actual).
4. Cantidad (El cual se generará aleatorio a no ser que el usuario ingrese dicha cantidad de 3 a 20 litros).
5. Categoría (premium corresponde al número 1, corriente corresponde al número 2 y eco corresponde al número 3. El cual se generará aleatorio a no ser que el usuario ingrese dicho número).
6. Método (Tcredito corresponde al número 1, Tdebito corresponde al número 2 y Efectivo corresponde al número 3. El cual se generará aleatorio a no ser que el usuario ingrese dicho número).
7. Numero de documento (número de 10 cifras que se generará aleatoriamente a menos que, el usuario lo digite).
8. Monto total (número el cual expresa la conversión de litro por su precio correspondiente).
9. Código surtidor (el cual dependiendo el constructor será generado por la clase surtidor o será escogido aleatoriamente, siendo un numero de 4 cifras).

Anexo:

Para decretar los precios individuales de las categorías de las gasolinas consideramos hacer un módulo llamado funciones, el cual contendrá una función llamada categoría gasolina, la cual actuará como a la red y será invocada por la clase ventas al momento de definir el monto total y hacer la conversión litros-dinero.

Siendo entonces la función categoría gasolina un arreglo [3][3] cuya primera dimensión define la región (0:Norte,1:centro,2:sur) y su segunda dimensión los precios del costo de 1 litro de gasolina según su categoría (0:premium.1:corriente,2:eco)

Sus métodos:

1. Obtener fecha hora actual, y es necesario agregar los siguientes hallazgos para hacer uso de las librerías <ctime> y <chrono>

El tipo real de now es std::chrono::system\_clock::time\_point, pero el usar auto evita tener que escribirlo explícitamente.

La función now() se utiliza para obtener el tiempo actual en un punto específico del tiempo. Es parte de la biblioteca <chrono>, que ofrece herramientas para trabajar con el tiempo y duraciones

Significado de now(): Obtener el tiempo actual: La función now() devuelve un objeto que representa el tiempo actual desde un punto de referencia (en general, desde el Epoch, que es el 1 de enero de 1970 a las 00:00:00 UTC). El resultado es un tipo de time\_point que indica el momento exacto en el tiempo en que se llamó a la función.

chrono::system\_clock::to\_time\_t(now): Esta función convierte el time\_point obtenido en un valor de tipo time\_t, que es una representación más tradicional del tiempo en C/C++ (esencialmente, el número de segundos transcurridos desde el 1 de enero de 1970, conocido como el Unix Epoch).

time\_t tiempoActual: Aquí se define una variable llamada tiempoActual de tipo time\_t que almacena el número de segundos desde el 1 de enero de 1970. Esta es la forma más básica de tiempo en muchos sistemas operativos.

tm\* partesTiempo = localtime(&tiempoActual);

localtime(&tiempoActual): Esta función toma un puntero a un valor time\_t (en este caso, &tiempoActual) y lo convierte en una estructura tm que descompone ese número de segundos en elementos legibles como el año, mes, día, horas, minutos, segundos, etc. La función localtime() toma el valor de tiempo universal (time\_t) y lo convierte a la hora local según la configuración regional del sistema.

tm\* partesTiempo: Aquí se define un puntero a una estructura tm. Esta estructura es la que contiene la fecha y la hora descompuesta en componentes individuales, como año, mes, día, hora, etc. Por ejemplo:

La estructura tm es parte de la biblioteca estándar de C/C++ y se utiliza para descomponer una representación de tiempo en sus componentes individuales, como el año, mes, día, hora, minutos, segundos, etc. Donde:

* tm\_min: Minutos (0-59).
* tm\_hour: Horas (0-23).
* tm\_mday: Día del mes (1-31).
* tm\_mon: Mes del año (0-11, donde 0 es enero y 11 es diciembre).
* tm\_year: Año desde 1900. Para obtener el año real, debes sumar 1900. Por ejemplo, si tm\_year es 124, el año actual es 2024.

1. Calcular monto actual, hace uso de la función mencionada anteriormente, la cual es una simple multiplicación entre la cantidad de litros y su cantidad según el arreglo basándose en la categoría y la region.
2. Constructores de ventas, en este caso posee dos ya que si el usuario desea hacer una simulación de venta con valores ingresados por el mismo y un código de surtidor aleatorio. El otro constructor esta pensado para cuando las clases se enlacen y cuyo argumento será el código del surtidor al cual pertenece la venta.
3. Mostrar información, como su nombre lo dice imprime en pantalla los valores de sus atributos basados en sus contructores.
4. Getters y setters.
   1. Creación clase surtidores: Esta clase tendrá los siguientes atributos:
5. Activo (actúa como una bandera la cual indica si a el surtidor se le acaban los 3 tipos de gasolina)
6. Código (Es un número de 4 cifras aleatorio)
7. Modelo (un carácter que va de la A-Z).
8. Ventas (la clase previamente creada).
9. Id de la estación (la cual será brindada por la clase tanque, y corresponde aun numero de 3 cifras.)
10. Litros disponibles (Será así mismo dado por la clase tanques y corresponde a la suma neta de los tres tipos de gasolina en el tanque, para así verificar posteriormente si el surtidor se encuentra activo o no).

Sus métodos:

1. Bool isactivo, este se encargará de mostrar la actividad o no delsurtidor.
2. Actualizar estado, este método se usará si el usuario decide cambiar la cantidad neta de combustible, ya que, si la cambia a 0, se desactivará el surtidor automáticamente.
3. Mostrar información.
4. Setters y getters.
5. Constructor, tendrá uno por defecto ya que era necesario al momento de crear un objeto en otra clase, y este cambiará sus valores cuando se contruya con el otro constructor que tiene por argumentos la cantidad total de litros que hay en el tanque, y el código de 3 cifras de la estación.
   1. Creación clase Tanque: Esta clase tendrá los siguientes atributos:
6. Eco (tipo de gasolina que almacenara su respectiva cantidad en litros).
7. Premium (tipo de gasolina que almacenara su respectiva cantidad en litros).
8. Corriente (tipo de gasolina que almacenara su respectiva cantidad en litros).
9. Estación (será el código de tres cifras de la estación a la cual corresponde el tanque y por lo tanto cada uno de sus surtidores, servirá para interconectar las clases).
10. Cantidad de surtidores (será un valor aleatorio el cual planteará cual es la cantidad de surtidores que posee dicho tanque) este servirá para decidir el tamaño del arreglo del siguiente numeral.
11. Un puntero de tipo Surtidores (clase previamente creada), la cual almacenara un arreglo dinámico con la información de los surtidores.

Sus métodos:

1. Contructor por defecto y constructor cuyo argumento es el código de 3 cifras de la estación que interconecta las clases para que haya correspondencia.
2. Getters y setters.
3. Método de impresión de atributos.
   1. Creación clase Estacion de Servicio: Esta clase tendrá los siguientes atributos:
4. Código (Número aleatorio de 4 cifras).
5. Gerente (Será constante para todas las estaciones ya que no es tan relevante).
6. Región (Norte corresponde al número 1, Centro corresponde al numero 2 y sur corresponde al numero 3 y será elegido aleatoriamente).
7. Arreglo estático de tipo float llamado gps (cuyas dos posiciones representarán la altitud y longitud respectivamente, serán calculadas aleatoriamente).
8. Islas (1-12 generadas aleatoriamente).
9. Tanque (clase previamente creada).

Sus métodos:

1. Un costructor sin argumentos.
2. Método para generar un arreglo de 2 datos para almacenar las coordenadas le latitud y longitud
3. Getters y setters
4. Método para imprimir atributos.