****

Entrega Final Proyecto de Clase

Programación 3

**Alumno:**

Edwin Noé Argueta Flores 32111584

**Sede de estudio:**

CEUTEC Centroamérica

**Docente:**

Oscar Fernando D’cuire Galeano

**Fecha de entrega:**

16 de Septiembre del 2024

**Link del video de mi programa**   
<https://youtu.be/9Px7wYd-tXM>

**Descripción del Caso a Desarrollar**

Gasolinera que ofrece diferentes tipos de combustible

El proyecto consiste en desarrollar un programa para una gasolinera que ofrecer diferentes tipos de combustible, este programa permitirá realizar gestiones CRUD para realizar los registros de la venta y compra de combustible.

**Descripción de Problemas a Resolver**

Además del CRUD debe resolverlos siguientes problemas:

* El programa no permitirá realizar la venta de combustible si la gasolinera no cuenta con la cantidad suficiente en inventario.
* Descuentos:
  + Basado en la edad del cliente o si el cliente tiene descuento aplicado, se otorgará un 25% de descuento automáticamente a los clientes que tengan la edad mayor o igual a 65 años (tercera edad).
  + Los clientes Regulares podrán acumular puntos, por cada compra acumulará acumulará el 10% en puntos los cuales solo se pueden canjear si los puntos son mayores a 100 puntos, cada punto de compra equivale a 1 lempira.
  + Los clientes corporativos **no** acumularan puntos sin embargo podrán tener la posibilidad de comprar crédito y pagar con crédito, por cada recarga de crédito que sea mayor a 1000 lps se le aplicara el 25% más al crédito, ej : si el cliente corporativo compra 1000 lps en crédito se le aplicara 1225 en crédito, en caso de que la compra sea menor a 1000 lps , únicamente se le aplicara 10%

**Definición de objetos**

**Combustible**:

Atributos:

* String nombreCombustible
* Double cantidadDisponible
* Int totalVentadGasolinaUnitaria
* Int totalVentasGasolinaGeneral (static)

Metodos:

* Combustible ();
* Combustible (String nombreCombustible , double cantidadCombustible, int totalVentaGasolinaUnitaria){}
* Getters and setters
* Void actualizarCantidad(double cantidadVendidaoComprada){}
* Void actualizaprecio(double nuevoprecio){}
* Void SumarVentaGasolinaGeneral(){totalVentasGasolinaGeneral= totalVentasGasolinaGeneral+1}
* Void SumarVentaGasolinaUnitaria(){totalVentasGasolinaGeneral= totalVentasGasolinaGeneral+1}
* Void imprimirGenelarinfo{}
* Void ActualizarInstanciArchivo( nombredecombustibe);

Descripcion:

* La clase Combustible facilita la administración de la información detallada sobre cada tipo de combustible disponible en la estación de servicio. Gestiona aspectos esenciales como el nombre del combustible, la cantidad actual combustible disponible de cada uno y las ventas realizadas específicamente para cada tipo de combustible. Además, mantiene un contador global para las ventas de combustibles, lo que ayuda en la clase ventas para realizar cálculos. Los métodos de la clase permiten actualizar el inventario disponible, registrar las transacciones de ventas y consultar la información almacenada que es necesaria para la clase factura además de eso tiene e método ActualizarInstanciArchivo el cual será de mucha ayuda para actual;izar archivo txt.

**Cliente:**

Atributos:

* int TotalClientes (static)
* int id
* String nombreCliente
* int edad
* double double descuento =25% (privado de tipo final)

Metodos:

* Cliente ();
* Cliente(String nombreCliente, int edad);
* Getters and setters (excepto por descuento)
* double calcularDescuentoterceraEdad(double cantidadGasolina) { if (edad >= 65) return cantidadGasolina \* descuento; else return cantidadGasolina; }
* CalcularDescuento() {sobreescribir para cliente coorporativo y regular }
* Void actualizarCliente(int id , string nombre , int edad){}
* Metodos CRUD {}
* Void Actualiizarcliente{}

Descripción:

La clase cliente gestionara la información necesaria de cada cliente para el correcto funcionamiento del programa, como la edad, nombre y descuento.

También tendrá los métodos CRUD para gestionar la instancia que se esta trabajando al igual que actualizar cual ayudara para actualizar la instancia dell archivo

Además, será la encargada de verificar si la persona tiene descuento para poder realizar los cálculos de descuento de tercera edad a los cuales son necesarios para la clase factura.

**Facturación:**

Atributos:

* Double totalganancia (static)
* Combustiblemasvendido (static)

Métodos:

* Facturacion ();
* Void ImprimirFactura(vector){cout<< impimefacura };
* Void verUltimafactura(){}
* Bool CalcularSiCuentaConCombustible(clase Combustible) ){return bool}
* string obtenerCombustibleMasVendido(){}

descripción:

* La clase Facturación gestiona la creación y almacenamiento de facturas en el sistema de la gasolinera. Almacena todas las facturas en un vector estático, permitiendo una manipulación fácil y centralizada de los datos. Además de imprimir y visualizar facturas, la clase cuenta con métodos y atributos estáticos para facilitar los cálculos de ganancias y el seguimiento del combustible más vendido.

**Clases Derivadas de tipo cliente** :

**Cliente :**

**Cliente Corporativo:**

Atributos:

* String nombreEmpresa
* Double CreditoDisponible;

Metodos:

* Getters and setters ;
* void aplicarCredito(double montoAplicar);
* void RecargarCredito(double montoComprar);

Descripción:

La clase cliente corporativo gestionara la información necesaria del crédito disponible del cliente corporativo obteniendo todos los beneficios planteados en la descripción del problema.

**Cliente:**

**Cliente Corporativo:**

Atributos:

* Double Puntos;

Metodos:

* Getters and setters ;
* void aplicarPuntos(double montoAplicar);
* double calcularCuantosPuntosAcumula(double montoComprado);

Descripción:

La clase cliente regular gestionara la información necesaria para aplicar los puntos y la acumulación de los puntos obteniendo todos los beneficios planteados en la descripción del problema.

**Implementación POO**

describa de brevemente de qué manera espera aplicar los principios de programación orientada a objetos en su proyecto.

**Abstracción** En el proyecto, se utilizarán la clase Cliente, Combustible y Facturación para modelar objetos del mundo real. Cada clase tienes aspectos diferentes.

**Encapsulamiento**: Los atributos de las clases estarán encapsulados, por lo que serán privados o protegidos dependiendo de la aplicación, y se accederá a ellos a través de métodos públicos (getters y setters) al igual que los métodos necesarios para el funcionamiento del programa.

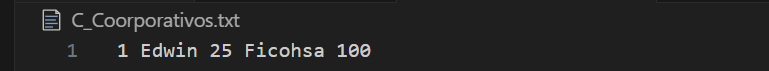
**Herencia**: La clase La clase Cliente actúa como clase base de la que se derivan ClienteCorporativo y ClienteRegular. Estas subclases heredan atributos y métodos comunes de Cliente, también incluyen funcionalidades adicionales, como RecargarCredito en ClienteCorporativo y calcularCuantosPuntosAcumula en ClienteRegular.

**Polimorfismo**: El polimorfismo se aplicara a métodos como Actuaizarcliente y Los Metodos CRUD , los cuales serán sobrescritos en las subclases ClienteCorporativo y ClienteRegular para proporcionar comportamientos específicos.

**Aspectos de diseño del proyecto**

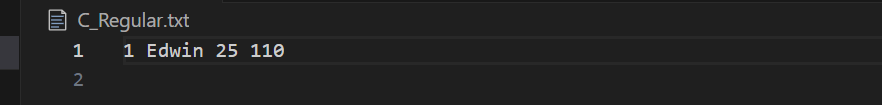
**C\_Coorporativos.txt:** Este Archivo se ubica en la misma carpeta que se encuentra el main y contiene ID, Nombre, edad, Nombre Empresa, Crédito Disponible todo en ese orden .

Ejemplo :

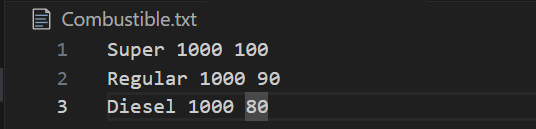
****

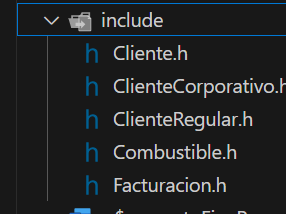
**C\_Regular.txt:**  Este Archivo se ubica en la misma carpeta que se encuentra el main y contiene ID, Nombre, edad y Crédito Disponible todo en ese orden.

Ejemplo:

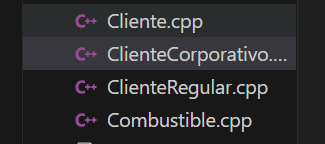


**Combustible.txt** Este Archivo se ubica en la misma carpeta que se encuentra el main y contiene el nombre de combustible, cantidad disponible y el precio de cada uno .   
Nota: este archivo se crea automáticamente con valores por defecto los cuales son :

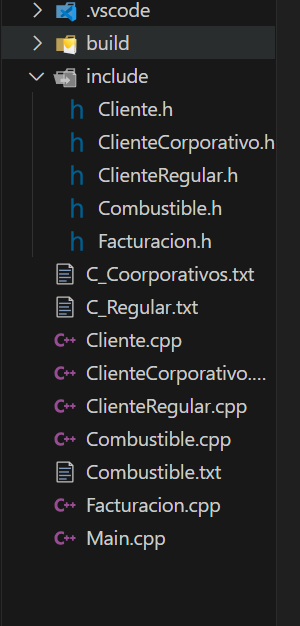


**Carpetas** include : Esta carpeta contiene todas as plantillas de cada case   


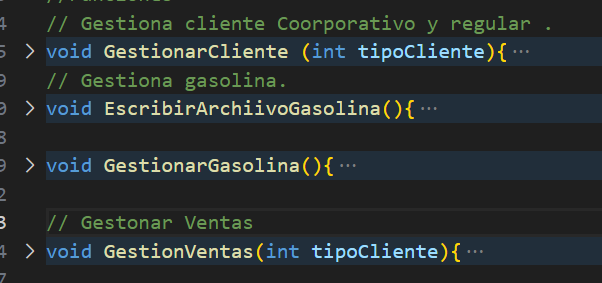
**Clases**: Las clases se encuentran fuera de esa carpeta y están situadas donde se encuentra el main .



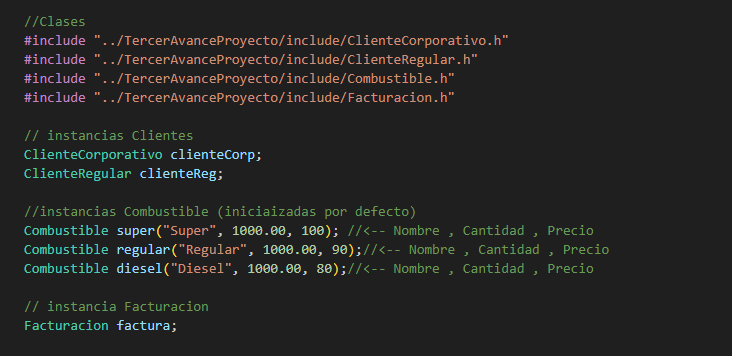
.

**Las carpetas y archivos** deberían de lucir así:   


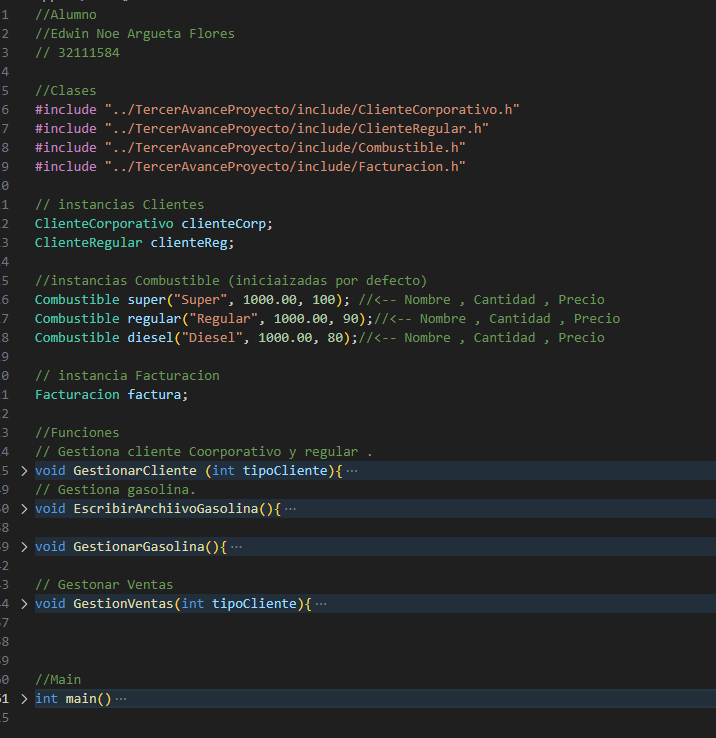
**Breve descripción del código:**

Para entender el funcionamiento del programa se comienza con el archivo main el cual contiene 3 funciones principales fuera del main :   


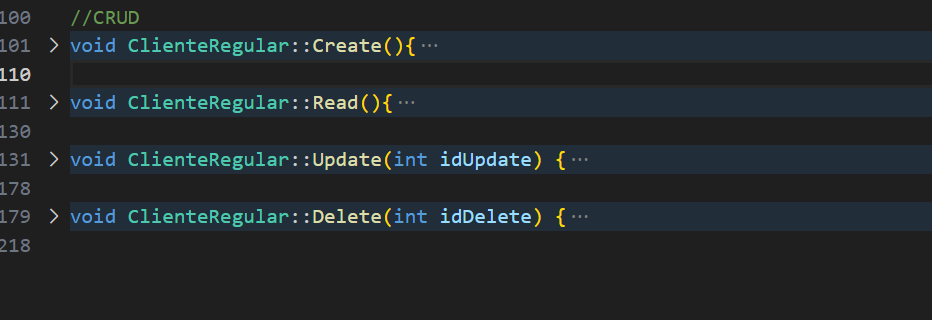
También incluye las clases he instancias a utilizar

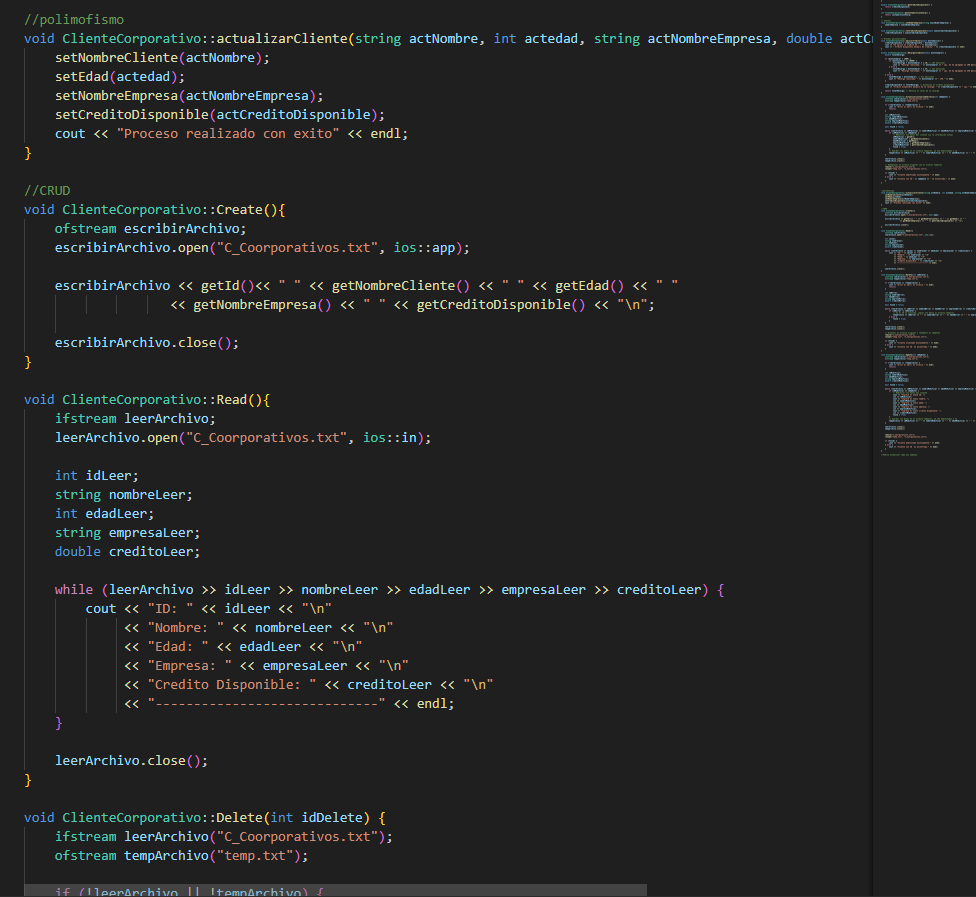


Por último el programa en total luce así :



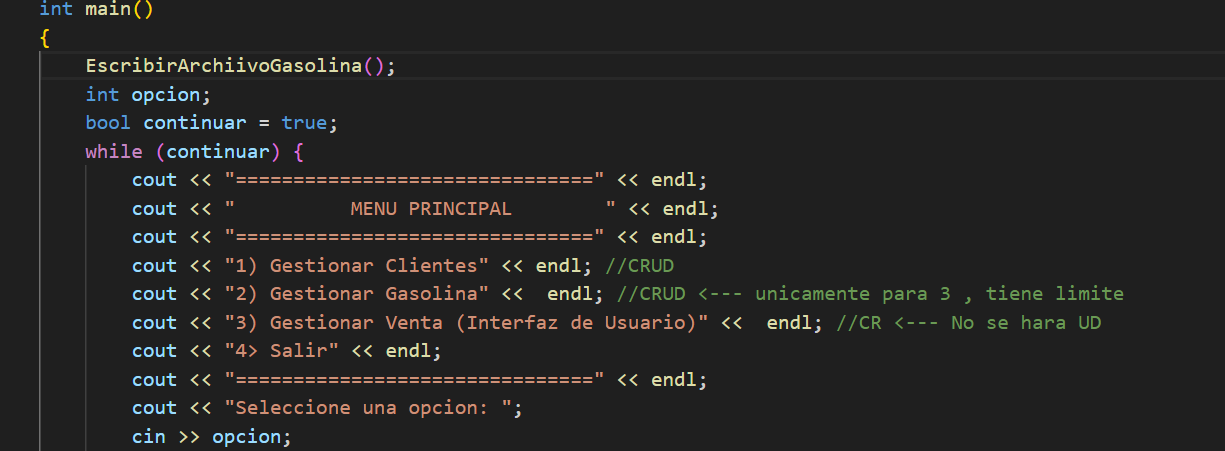
* Gestionar Clientes: Esta función contiene un menú para gestionar todos los usuarios pudiéndonos permitir realizar un CRUD y estas se gestionan mediante los cases de **ClienteCoorporativo** y **CienteRegular** las cuales tiene los siguientes métodos heredados y haciendo polimorfismo de la clase **Cliente** la cual se encarga de gestionar cada archivo, cada case tanto corporativo como regular lucen así :





* Gestionar Gasolina y Escribir Archivos Gasolina: ambas funciones son encargadas de gestionar las as gasolinas, en especia gestionar gasolina en la cual permite realizar un CRUD y EscribirArchiivoGasolina crea el archivo para gestionar gasolina inicial al ejecutar el main y al igual que el **Cliente** tiene sus respectivos métodos .
* GestionarVentas : esta función muestra un menú del cliente en general , permite realizar las compras de gasolina a nivel de usuario y de esa manera resolver los problemas a resolver planteados en e programa.

Estas funciones se ejecutan en el main y pide la función a realizar dependiendo de la opción seccionada.



|  |  |  |
| --- | --- | --- |
| Fecha | Actividad | Duración |
| Jueves 15 de agosto | Creación del diagrama CRC (versión 1) y descripción del caso. | 2 horas |
| Viernes 16 de agosto | Replanteamiento de casos a resolver para aplicar POO y redefinición del diagrama CRC (versión 2). | 2 horas |
| Sábado 17 de agosto | Redefinición de objetos y ajuste del diagrama CRC (versión 3); implementación de clases heredadas de Cliente y definición de problemas a resolver. | 3 horas |
| Domingo 18 de agosto | Revisión del proyecto, ajustes en las clases y descripciones; implementación de la bitácora. | 1hora y 30 min |
| Martes 3 de septiembre | Creación de clases del proyecto usando como base el primer avance versión 1 | 1 hora y 30 minG |
| Miércoles 4 de septiembre | Correr los métodos hechos y corrección de pequeños ajustes . | 45 min |
| Jueves 12 de septiembre | Separación de código a clases y plantilla separadas | 2 horas |
| Jueves 13 de septiembre | Creación de main y funciones Gestionarcliientes , EscribirArchivoGasoina y GestionarGasoina | 2 horas |
| Jueves 14 de septiembre | Debugeo de código en general . | 1 horas y media |
| Jueves 15 de septiembre | Creacion de función GestionVentas y debugeo de código | 4 horas |
| Jueves 16 de septiembre | Creación de bitácora y video | 2 horas |

**Dificultades encontradas en el desarrollo del proyecto**

* Legibilidad de código: A medida el proyecto se iba realizando en el archivo .cpp era muy difícil leer el código sin embargo a través desplantillos y funciones aparte se pudo resolver esa dificultad.
* Abstracción y modificaciones de clases: Los avances 1 y 2 fueron de mucha ayuda al poder tener una idea clara sin embargo al momento de desarrollar el programa se vio la necesidad de modificar ciertas clases y poder abstraer cada una de sus responsabilidades en el proyecto, por ejemplo, los métodos CRUD dentro de la case cliente, al final se pudo resolver pensando y creado los nuevos métodos incorporándolos de manera exitosa .
* Lógica de programación: Tuve dificultades de lógica en ciertas iteraciones del programa al igual que no sabia como exactamente funcionarían las instancias y mandarlas directamente al archivo .txt correspondiente para hacer el crud , al final se superó el obstáculo realizando un método para poder actualizar la clase y que el solo se mande al archivo txt dependiendo de lo que se quiera realizar .
* Gestión de tiempo: la planificación de dividir la elaboración del proyecto, en programación es difícil manejar el tiempo de manera efectiva y más por la curva de aprendizaje. En el proyecto se priorizo lo que contenía la rúbrica modificando algunos detalles de proyecto.