Imagen que contiene Texto

Descripción generada automáticamenteNombre: Jennifer Rebeca García Moncada Ced: 115110687



**Clase Cliente**

**Atributos:**

nombre (String) (nombre del cliente) (Para identificar y reconocer cada cliente)

dirección (String) (dirección del cliente) (Para enviar notificaciones, facturas o algún servicio)

teléfono (string) (numero de teléfono) (Para confirmaciones o recordatorios de las citas)

correo\_electronico (string) (dirección de correo del cliente) (Para enviar notificaciones, cotizaciones, facturas o alguna comunicación oficial)

**Justificación de la clase:** La información personal de los clientes del taller se almacena en esta clase, lo que es esencial para cualquier tipo de comunicación y seguimiento de servicios.

**Clase Vehículo**

**Atributos:**

Marca: (string) (marca el vehículo) (Para identificar el fabricante del vehículo)

Modelo: (string) (modelo del vehículo) (Para especificar el tipo de vehículo)

Año: (Int) (año que se fabricó el vehículo) (Para identificar la antigüedad del vehículo)

VIN: (string) (id de vehículo) (identificador único del vehículo)

Propietario: (hace referencia a la clase cliente) (Para asociar el vehículo con su dueño)

**Justificación de la clase:** Llevar un registro detallado de cada vehículo que ingresa al taller es fundamental para poder rastrear su historial de servicios y propietarios.

**Clase Servicio**

**Atributos:**

Nombre: (string) (nombre del servicio) (Para diferenciar los distintos servicios)

Descripción: (string) (descripción del servicio) (Proporciona una descripción clara de cada servicio)

Costo: (double) (costo del servicio) (Para cotizar y gestionar los pagos por los servicios brindados)

**Justificación de la clase:** Para facilitar la cotización y facturación, así como para asignar el trabajo adecuado a los mecánicos, cada tipo de servicio que ofrece el taller debe estar clara mente definido y registrado.

**Clase Cita**

**Atributos:**

Cliente: (Cliente) (referencia a la clase cliente) (Para saber quien solicitó el servicio)

Vehículo: (Vehículo) (referencia a la clase vehículo) (Para identificar el vehículo que recibe el servicio)

Servicio: (Servicio) (referencia a la clase servicio) (Para especificar el trabajo que se realiza durante la cita)

fecha\_hora: (datetime) (fecha y hora de la cita) (Para programar y gestionar el tiempo del servicio)

mecánico: (mecánico) (referencia a la clase mecánico) (Para asignar la tarea al mecánico)

**Justificación de la clase:** Para organizar el flujo de trabajo en el taller, la gestión de citas es importante para que cada servicio se realice en un tiempo adecuado.

**Clase Mecánico**

**Atributos:**

Nombre (string) (nombre del mecánico) (Para identificar el mecánico)

Especialidad: (string) (Especialidad o área del mecánico) (Para asignar las tareas específicas del mecánico)

Disponibilidad: (List of datetime) (Lista de fechas y horas de la disponibilidad del mecánico) (Para programar las citas de manera eficiente).

**Justificación de la clase:** Es necesario tener la información de cada mecánico para asignar servicios de acuerdo con la especialidad de cada mecánico y la disponibilidad de cada uno.



**Clase Cliente**

**Métodos:**

**Init (self, nombre, dirección, teléfono, correo)**

Constructor para inicializar un objeto cliente con los atributos.

**Actualizar\_direccion (self, nueva\_direccion)**

Actualiza la dirección del cliente

**Actualizar\_telefono(self, nuevo\_telefono)**

Actualiza el teléfono del cliente

Actualizar\_correo(self, nuevo corre)

Actualiza el correo del cliente

**Obtener\_informacion(self)**

Obtener toda la información del cliente en formato de diccionario o string

**Clase Vehículo**

Init (self, marca, modelo, año, vin, propietario)

Constructor para inicializar un objeto vehículo.

**Actualizar\_propietario (self, nueva\_propietario)**

Actualiza el propietario del vehículo

**Obtener\_informacion(self)**

Método para obtener toda la información del vehículo.

Clase Servicio

Init(self, nombre, descripción, costo)

Constructor para inicializar un objeto servicio.

**Actualizar\_descripcion (self, nueva\_descripcion)**

Actualiza la descripción del servicio.

**Actualizar\_costo (self, nuevo\_costo)**

Actualiza el costo del servicio.

**Obtener\_informacion(self)**

Método para obtener toda la información del servicio.

**Clase Cita**

**Init(self, cliente, vehículo, servicio, fecha\_hora, mecánico)**

Constructor para inicializar un objeto cita

**Actualizar\_fecha\_hora (self, nueva\_fecha\_hora)**

Actualiza la fecha y hora de la cita

**Actualizar\_mecanico (self, nuevo\_mecanico)**

Actualiza el mecánico asignado a la cita

**Obtener\_informacion(self)**

Método para obtener toda la información de la cita.

Clase Mecánico

**Init(self, nombre, especialidad, disponibilidad)**

Constructor para inicializar un objeto mecánico

**Agregar\_disponibilidad(self, nueva\_disponibilidad)**

Agrega una nueva disponibilidad a la lista de mecánicos

**Eliminar\_disponibilidad(self, disponibilidad\_eliminar)**

método para eliminar una disponibilidad especifica.

**Actualizar\_especialidad(self, nueva\_especialidad)**

Actualiza la especialidad del mecánico.

**Obtener\_informacion(self)**

Método para obtener toda la información del mecanico.

Texto

Descripción generada automáticamente

Se elige la de Gestión Clientes

Justificación: La implementación de Gestión Clientes mejora la eficiencia y la calidad del servicio del taller, también contribuye la satisfacción y la retención de clientes.

**Pantalla Gestión Cliente**

**Espacios de Nombres y Clase principal**

‘namespace AutoSoluciones’: define un espacio de nombre para organizar el código

‘class Program’: Punto de entrada del programa

Texto

Descripción generada automáticamente

**Declaración de Variables y Menú Principal**

Cliente[] clientes = new Cliente[100]; Arreglo para almacenar hasta 100 clientes

int clienteCuenta = 0; Contador de clientes registrados

int opcion; Almacena la opción seleccionada por el usuario.

El bucle do-while muestra el menú y solicita al usuario que escoja una opción.

Texto

Descripción generada automáticamente

**Manejo de Opciones del Menú**

**Caso 1:** Agregar cliente: llama al método AgregarCliente y aumenta el contador.

**Caso 2: Modificar cliente:** llama el método ModificarCliente para modificar la información del cliente.

**Caso 3: Eliminar cliente:** llama el método EliminarCliente y ajusta el contador.

**Caso 4: Mostrar cliente:** llama el método MostrarClientes para mostrar los clientes.

**Caso 5: Salir:** sale del bucle y termina el programa.

**Default:** Maneja las opciones no validas.

Texto

Descripción generada automáticamente

**Método para Gestionar Clientes**

Solicita los datos del cliente nuevo, los almacena en el objeto cliente y retorna.

**Texto

Descripción generada automáticamente**

**Método ModificarCliente**

Solicita el ID del cliente a modificar, lo verifica y permite modificar los datos.

**Texto

Descripción generada automáticamente**

**Método EliminarCliente**

Solicita el ID del cliente a eliminar, lo verifica y reorganiza el arreglo para eliminar el cliente escogido.

**Texto

Descripción generada automáticamente**

**Método MostrarClientes**

Recorre el arreglo y muestra la información de cada cliente registrado.

Texto

Descripción generada automáticamente

**Clase Cliente**

Define las propiedades del cliente: nombre, dirección, teléfono y correo.

Texto

Descripción generada automáticamente con confianza media