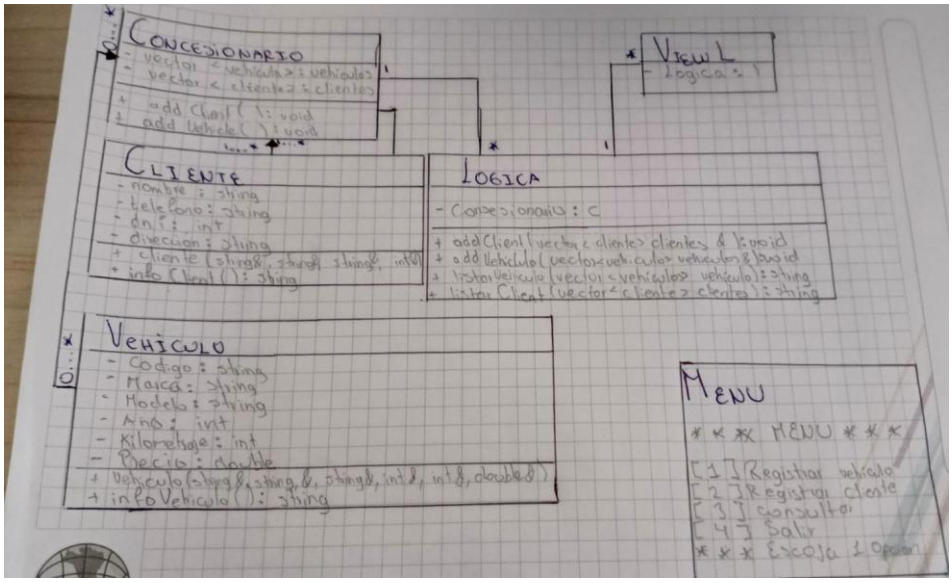
Guía de Práctica de aplicación y experimentación de los aprendizajes de la Universidad Politécnica Salesiana	
Carrera:	**CIENCIAS DE LA COMPUTACIÓN**
Nivel:	**2do**
Asignatura:	**PROGRAMACIÓN ORIENTADA A OBJETOS**
Desarrollado por:	**Johan Peralvo**
Grupo:	**N°4**
Resultados de Aprendizaje:	**Construye programas utilizando el paradigma de programación orientada a objetos.**
Indicador de logro:	**Utiliza el patrón MVC para el desarrollo de aplicaciones de software.**
Práctica/Deber Número:	**Practica N°5**
Horas Dedicadas:	**2 y media**

DESCRIPCIÓN DE LA PRÁCTICA:

Analice las clases creadas en la práctica e identifique cuales son los atributos y métodos, y construya el diagrama de clases con sus correspondientes clases, si es un algoritmo coloque su flujograma correspondiente.



1.1 En base al diagrama de clases generado/algoritmo, construir la aplicación

CLASE: `CLIENTE_H`

```
#ifndef CLIENTE_H
#define CLIENTE_H
#include <string>
#include <iostream>
```

```
class Cliente {
private:
   std::string dni;
   std::string nombre;
   std::string telefono;
   std::string direccion;

public:
   Cliente(const std::string& dni, const std::string& nombre, const std::string&
telefono, const std::string& direccion);

   const std::string& getDni() const { return dni; }
   const std::string& getNombre() const { return nombre; }
   const std::string& getTelefono() const { return telefono; }
   const std::string& getDireccion() const { return direccion; }

   void mostrarCliente() const;
};

#endif // CLIENTE_H
```

CLASE: CONTROLADOR_H

```
#ifndef CONTROLADOR_H
#define CONTROLADOR_H
#include <vector>
#include <string>
#include "vehiculo.h"
#include "cliente.h"

class Controlador {
private:
   std::vector<Vehiculo> vehiculos;
   std::vector<Cliente> clientes;

public:
   void registrarVehiculo(const Vehiculo& vehiculo);
   void registrarCliente(const Cliente& cliente);

   void listarVehiculos() const;
   void listarClientes() const;
};

#endif // CONTROLADOR_H
```

CLASE: VEHICULO_H

```
    #ifndef VEHICULO_H
#define VEHICULO_H
#include <string>
#include <iostream>
```

```
class Vehiculo {
private:
   std::string codigo;
   std::string marca;
   std::string modelo;
   int anio;
   int kilometraje;
   double precio;

public:
   Vehiculo(const std::string& codigo, const std::string& marca, const std::string&
modelo, int anio, int kilometraje, double precio);

   const std::string& getCodigo() const { return codigo; }
   const std::string& getMarca() const { return marca; }
   const std::string& getModelo() const { return modelo; }
   int getAnio() const { return anio; }
   int getKilometraje() const { return kilometraje; }
   double getPrecio() const { return precio; }

   void mostrarVehiculo() const;
};

#endif // VEHICULO_H
```

CLASE: CLIENTE_H

```
#include "headers/cliente.h"

Cliente::Cliente(const std::string& dni, const std::string& nombre, const std::string&
telefono, const std::string& direccion)
   : dni(dni), nombre(nombre), telefono(telefono), direccion(direccion) {}

void Cliente::mostrarCliente() const {
   std::cout << "  DNI: " << dni << "\n"
         << "  Nombre: " << nombre << "\n"
         << "  Telefono: " << telefono << "\n"
         << "  Direccion: " << direccion << std::endl;
}
```

CLASE: CONTROLADOR_H

```
#include "headers/controlador.h"
#include <iostream>

void Controlador::registrarVehiculo(const Vehiculo& vehiculo) {
   vehiculos.push_back(vehiculo);
}

void Controlador::registrarCliente(const Cliente& cliente) {
   clientes.push_back(cliente);
```

```
}

void Controlador::listarVehiculos() const {
    std::cout << "\n--- Lista de Vehiculos ---\n";
    if (vehiculos.empty()) {
        std::cout << "No hay vehiculos registrados.\n";
    } else {
        for (const auto& vehiculo : vehiculos) {
            vehiculo.mostrarVehiculo();
            std::cout << "-------------------------\n";
        }
    }
}

void Controlador::listarClientes() const {
    std::cout << "\n--- Lista de Clientes ---\n";
    if (clientes.empty()) {
        std::cout << "No hay clientes registrados.\n";
    } else {
        for (const auto& cliente : clientes) {
            cliente.mostrarCliente();
            std::cout << "-------------------------\n";
        }
    }
}
```

CLASE: VEHICULO_H

```
#include "headers/vehiculo.h"

Vehiculo::Vehiculo(const std::string& codigo, const std::string& marca, const
std::string& modelo, int anio, int kilometraje, double precio)
    : codigo(codigo), marca(marca), modelo(modelo), anio(anio),
kilometraje(kilometraje), precio(precio) {}

void Vehiculo::mostrarVehiculo() const {
    std::cout << "  Codigo: " << codigo << "\n"
              << "  Marca: " << marca << "\n"
              << "  Modelo: " << modelo << "\n"
              << "  Anio: " << anio << "\n"
              << "  Kilometraje: " << kilometraje << "\n"
              << "  Precio: " << precio << std::endl;
}
```

1.2 Generar una clase main que cumpla con los siguientes requisitos planteados en el problema:

Método: main	Capturas de Pantalla con cada opción ejecutada

```
#include <iostream>
#include <string>
#include "headers/controlador.h"

using namespace std;

int main() {
   Controlador controlador;
   int opcion;

   do {
      cout << "\n--- Menu de la Concesionaria ---\n";
      cout << "1. Registrar vehiculo\n";
      cout << "2. Registrar cliente\n";
      cout << "3. Consultar listados\n";
      cout << "4. Salir\n";
      cout << "Seleccione una opcion: ";
      cin >> opcion;
      cin.ignore(); // Limpiar el newline del buffer

      if (opcion == 1) {
         string codigo, marca, modelo;
         int anio, kilometraje;
         double precio;

         cout << "Ingrese codigo del vehiculo: ";
getline(cin, codigo);
         cout << "Ingrese marca del vehiculo: ";
getline(cin, marca);
         cout << "Ingrese modelo del vehiculo: ";
getline(cin, modelo);
         cout << "Ingrese anio del vehiculo: "; cin >> anio;
         cout << "Ingrese kilometraje del vehiculo: "; cin
>> kilometraje;
         cout << "Ingrese precio del vehiculo: "; cin >>
precio;
         cin.ignore();

         Vehiculo nuevoVehiculo(codigo, marca, modelo,
anio, kilometraje, precio);
         controlador.registrarVehiculo(nuevoVehiculo);
         cout << "Vehiculo registrado con exito.\n";

      } else if (opcion == 2) {
```

```
--- Menu de la Concesionaria ---
1. Registrar vehiculo
2. Registrar cliente
3. Consultar listados
4. Salir
Seleccione una opcion:
```

```
Seleccione una opcion: 1
Ingrese codigo del vehiculo: 12223
Ingrese marca del vehiculo: Honda
Ingrese modelo del vehiculo: Cherry
Ingrese anio del vehiculo: 200
Ingrese kilometraje del vehiculo: 0
Ingrese precio del vehiculo: 120000
Vehiculo registrado con exito.
```

```
Seleccione una opcion: 2
Ingrese DNI del cliente: 1589657852
Ingrese nombre del cliente: Juan
Ingrese telefono del cliente: 0236548941
Ingrese direccion del cliente: Bretaña
Cliente registrado con exito.
```

```
-- Menu de la Concesionaria ---
. Registrar vehiculo
. Registrar cliente
. Consultar listados
. Salir
eleccione una opcion: 3

-- Que desea consultar? ---
. Listar todos los vehiculos
. Listar todos los clientes
eleccione una opcion:
```

```
-- Menu de la Concesionaria ---
. Registrar vehiculo
. Registrar cliente
. Consultar listados
. Salir
eleccione una opcion: 4
aliendo del sistema. Hasta luego!
rocess exited with code: 0.
```

```
            string dni, nombre, telefono, direccion;

            cout << "Ingrese DNI del cliente: "; getline(cin,
        dni);
            cout << "Ingrese nombre del cliente: ";
        getline(cin, nombre);
            cout << "Ingrese telefono del cliente: ";
        getline(cin, telefono);
            cout << "Ingrese direccion del cliente: ";
        getline(cin, direccion);

            Cliente nuevoCliente(dni, nombre, telefono,
        direccion);
            controlador.registrarCliente(nuevoCliente);
            cout << "Cliente registrado con exito.\n";

        } else if (opcion == 3) {
            int consultaOpcion;
            cout << "\n--- Que desea consultar? ---\n";
            cout << "1. Listar todos los vehiculos\n";
            cout << "2. Listar todos los clientes\n";
            cout << "Seleccione una opcion: ";
            cin >> consultaOpcion;
            cin.ignore();

            if (consultaOpcion == 1) {
                controlador.listarVehiculos();
            } else if (consultaOpcion == 2) {
                controlador.listarClientes();
            } else {
                cout << "Opcion de consulta invalida.\n";
            }

        } else if (opcion == 4) {
            cout << "Saliendo del sistema. Hasta luego!\n";
        } else {
            cout << "Opcion invalida. Por favor, intente de
        nuevo.\n";
        }

    } while (opcion != 4);

    return 0;
}
```

Problemas detectados durante el desarrollo de la práctica/deber