**Mini proyectó**

Hugo Javier Ledesma

María José Ortiz Sinisterra

Camilo Abel Sol Mosquera.

noviembre 2023

Institución Universitaria Antonio José Camacho.

Valle del cauca.

Ingeniería de software ll

Patrón de diseño (ESTRUCTURAL):

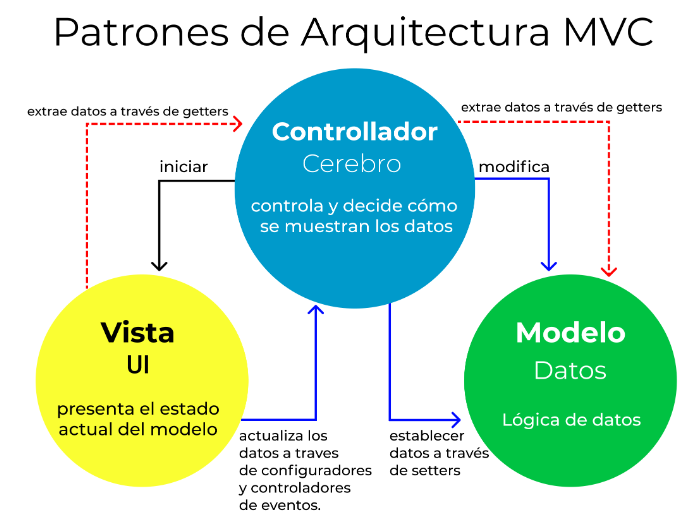
El patrón de diseño escogido por mi para la realización del miniproyecto fue el MVC (Modelo Vista Controlador)

Teniendo en cuenta que el MVC es un patrón de diseño muy conocido en el mundo de la programación web, me permite ajustarme de manera más coda, ya que es un patrón que ya sea directa o indirectamente he usado en proyectos web anterior y actual mente, pero ¿qué es MVC (Modelo Vista Controlador)?

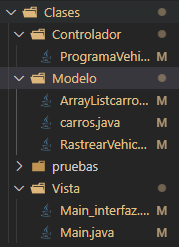
MVC (Modelo Vista Controlador)

MVC significa modelo (model) vista (view) controlador (controller). Esto es lo que significan cada uno de esos componentes.

* Modelo: El backend que contiene toda la lógica de datos
* Vista: El frontend o interfaz gráfica de usuario (GUI)
* Controlador: El cerebro de la aplicación que controla como se muestran los datos.



Ahora, este patrón esta aplicado de la siguiente manera:

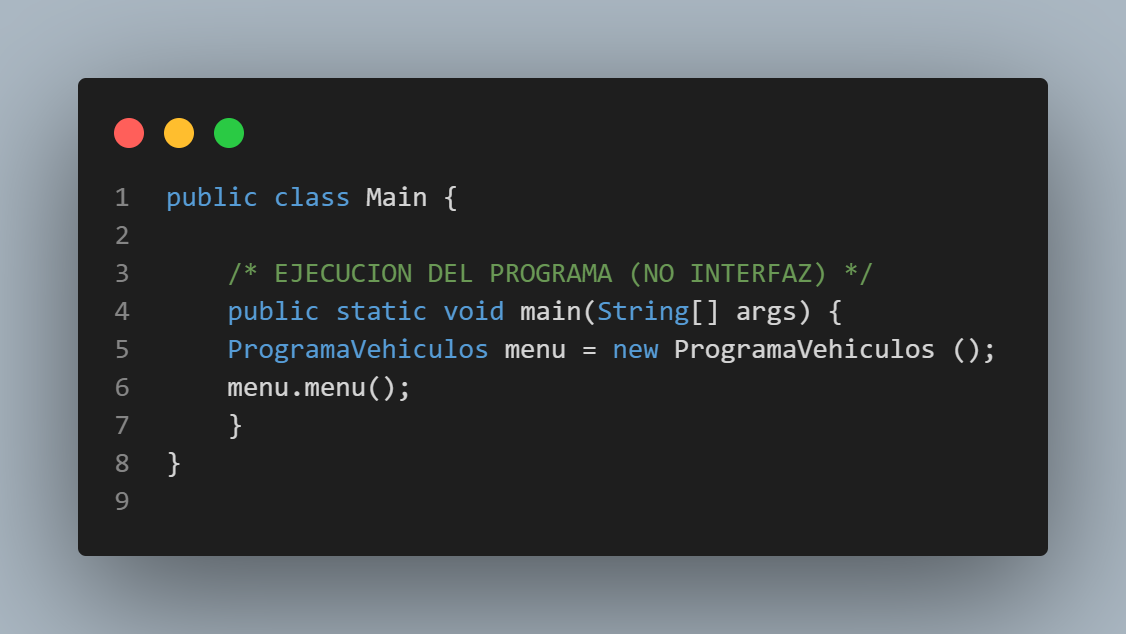
Podemos ver que contamos con carpetas separadas según el rol que cumple cada una de las clases del proyecto donde:

**Modelo:** Las clases ArrayListcarros, carros, RastrearVehiculo

**Vista:** La vista no está claramente definida en el código proporcionado. Sin embargo, en una aplicación de escritorio basada en Java Swing, la vista podría estar compuesta por ventanas y componentes de interfaz de usuario por ende nuestra vista será la clase Main (para la versión no interfaz) y Main\_interfaz (para la versión con interfaz).

**Controlador:** La clase ProgramaVehiculos se encarga de la lógica de la aplicación y la interacción con el usuario, por lo que podría considerarse el controlador principal.

Código de clase de ejecución (no interfaz – no patrón):



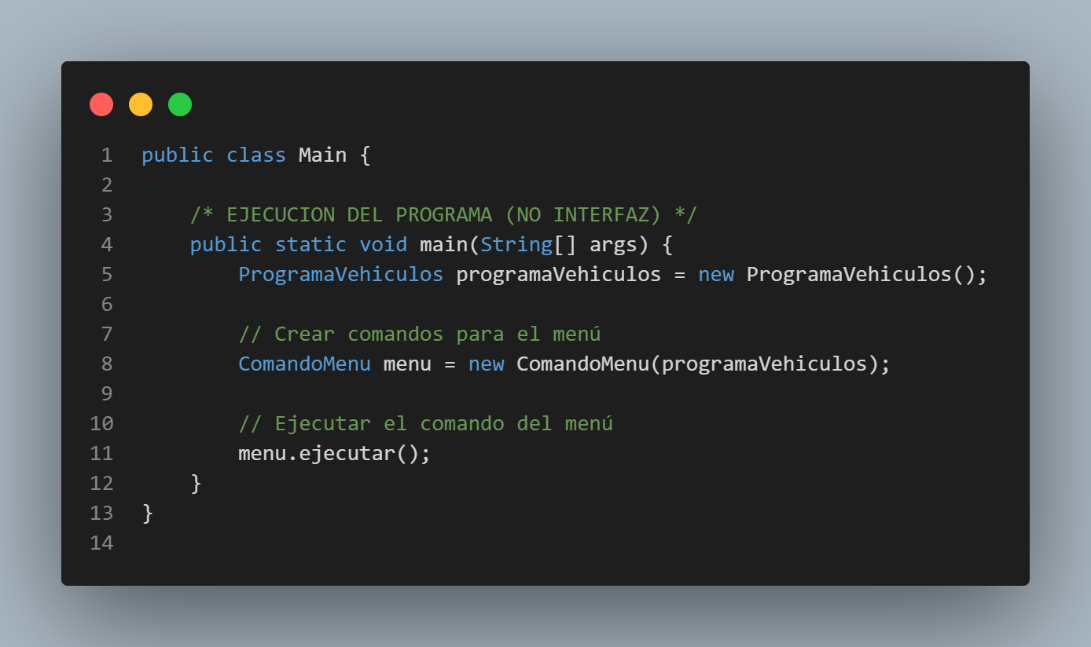
Código de clase de ejecución (Interfaz – no patrón):

Texto

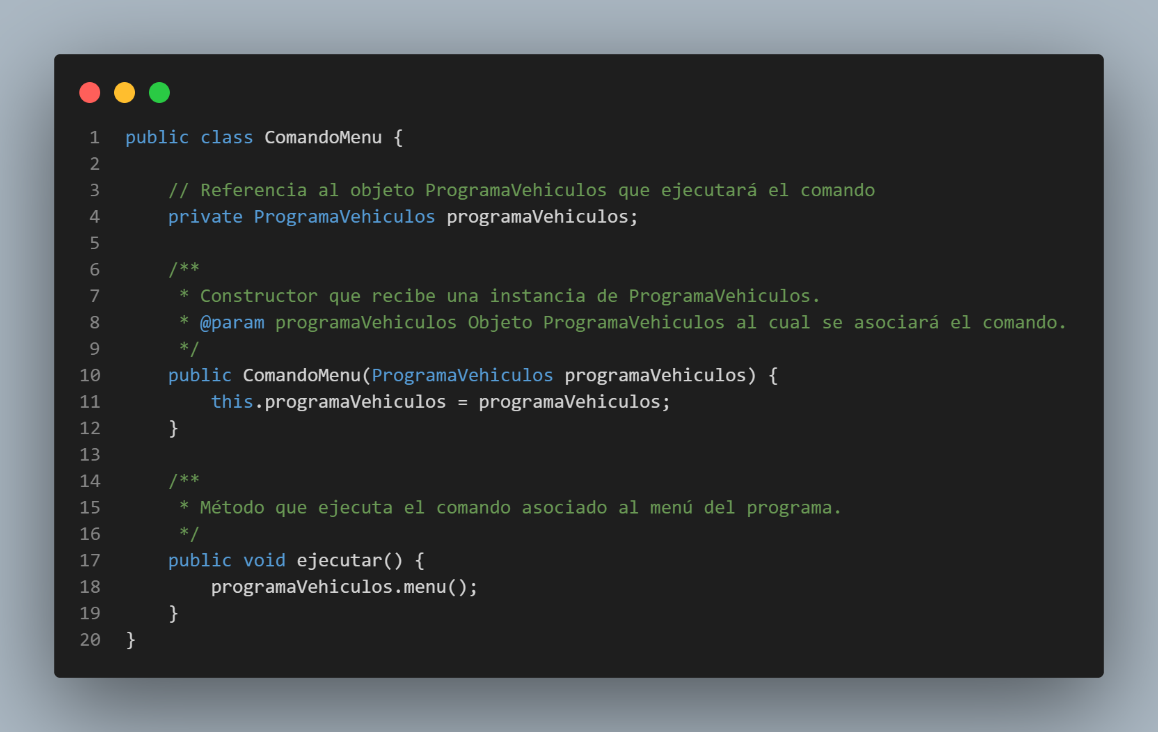
Descripción generada automáticamente

Patrón de diseño (CREACIONAL):

"Patrón de Diseño de Comando". Este patrón se utiliza para encapsular una solicitud como un objeto, lo que permite la parametrización de los clientes con distintas solicitudes, encolamiento de las solicitudes y la posibilidad de soportar operaciones que pueden deshacerse.



Aquí, se ha creado una clase ComandoMenu que encapsula la operación de ejecutar el menú del programa. Esta clase puede tener métodos adicionales o parámetros según sea necesario.



Este patrón puede ser útil si, en el futuro, necesitas extender tu programa para manejar diferentes comandos o si quieres implementar características como deshacer/rehacer. También hace que el código sea más flexible y fácil de mantener.

Diagrama de clases:

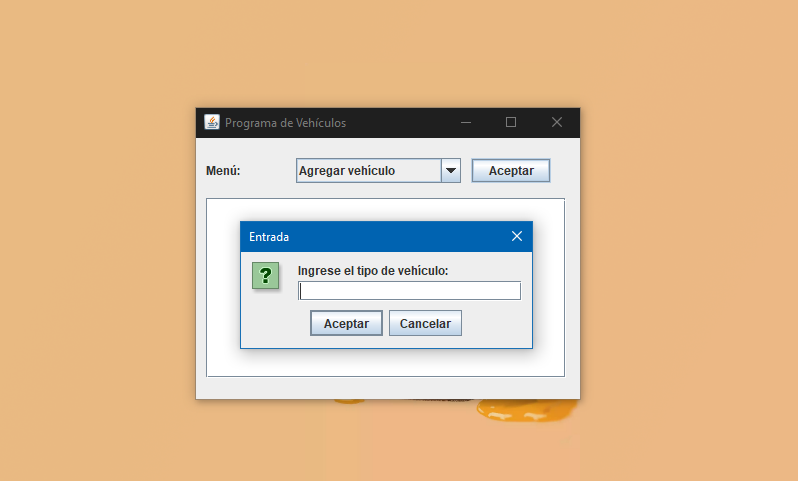
…………………..

Instrucciones y requerimientos cumplidos:

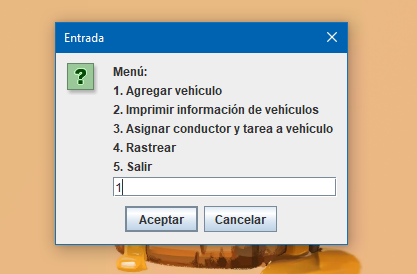
* **El sistema debe permitir la incorporación de nuevos tipos de vehículos y sensores de forma sencilla y extensible.**

**Demostración :**

Interfaz:



**No Interfaz:**

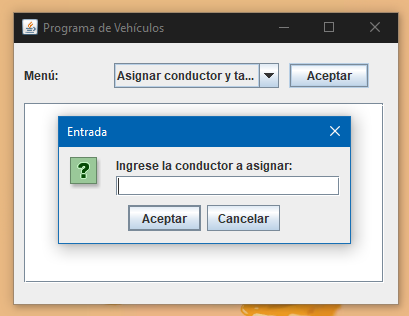
Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

* **Debe asignar de manera eficiente tareas a conductores y vehículos disponibles, considerando prioridades y capacidades.**

**Demostración :**

Interfaz:



**No Interfaz:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamente

* Debe permitir el monitoreo en tiempo real de la ubicación de los vehículos, sus estados y rutas.

**Demostración :**

Interfaz:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**No Interfaz:**

Interfaz de usuario gráfica

Descripción generada automáticamente

* El sistema debe ser escalable para soportar una flota en crecimiento y futuras expansiones.

🧐