Imagen que contiene Icono

Descripción generada automáticamente

**DESARROLLO DE SOFTWARE**

**GUÍA 2 – ACTIVIDAD 2**

**JAVA E INTERFACES GRAFICAS**

**AUTORES:**

MAYCKOLL ANDRÉS TORRES DIAZ.

**TUTOR**

DILSA ENITH TRIANA MARTÍNEZ

**3 DE MARZO DE 2024**

**INTRODUCCIÓN**Este proyecto tiene como objetivo la creación de una aplicación de escritorio en Java, que sea capaz de calcular los impuestos de un vehículo con base en esos factores.

El sistema utiliza Swing como tecnología para construir la interfaz gráfica, y sigue la arquitectura Modelo Vista Controlador (MVC), lo que asegura que el modelo de negocio sobre cálculo de impuestos esté separado de la lógica de la interfaz de usuario, mejorando la escalabilidad, mantenibilidad y flexibilidad del sistema.

El modelo de vehículo contiene las características básicas de un automóvil, como su marca, modelo, año de fabricación, cilindraje, avalúo y tipo de uso. El controlador gestiona los eventos de la interfaz de usuario y coordina la interacción entre la vista y el modelo. Finalmente, la vista es la parte visual que interactúa con el usuario, proporcionando formularios para ingresar los datos y mostrar los resultados.

**OBJETIVOS**

**Objetivo General:**

* Desarrollar un sistema de gestión en Java que permita calcular el impuesto sobre vehículos en función de su marca, modelo, año de fabricación, cilindraje, valor de avalúo y tipo de uso (particular o público), implementando un sistema con arquitectura Modelo-Vista-Controlador (MVC).

**Objetivos Específicos:**

* Diseñar una interfaz gráfica amigable y funcional utilizando Swing, que permita al usuario ingresar los datos del vehículo.
* Implementar la lógica de cálculo de impuestos teniendo en cuenta factores como la antigüedad del vehículo, cilindraje y tipo de uso, siguiendo las normativas fiscales de la aplicación.

Mostrar el resultado del cálculo del impuesto en tiempo real a través de la interfaz gráfica, garantizando una experiencia de usuario fluida y directa.

**RESOLUCIÓN DE LA PROBLEMÁTICA**El sistema fue implementado utilizando el lenguaje de programación Java, y sigue una arquitectura orientada a objetos y MVC.   
  
El sistema utiliza Swing como tecnología para construir la interfaz gráfica, y sigue la arquitectura Modelo-Vista-Controlador (MVC), lo que asegura que el modelo de negocio (cálculo de impuestos) esté separado de la lógica de la interfaz de usuario, mejorando la escalabilidad, mantenibilidad y flexibilidad del sistema.  
  
 **Tabla de Clases Gestión Parqueadero**  
  
En esta tabla se distinguen todos aquellos métodos y atributos que se deben crear para cada una de las clases, ya que es necesario para que el sistema funcione adecuadamente.

|  |  |  |
| --- | --- | --- |
| Clase | Atributos | Metodos |
| Vehiculo | - String marca  - String modelo  - int anoFabricacion  - int cilindraje  - double avaluoComercial  - String tipoUso | + getMarca(): String  + getModelo(): String  + getAnoFabricacion(): int  + getCilindraje(): int  + getAvaluoComercial(): double  + getTipoUso(): String |
| CalculadoraImpuestos |  | + calcularImpuesto(vehiculo: Vehiculo): double |
| Vista | - JTextField txtMarca  - JTextField txtModelo  - JTextField txtAnoFabricacion  - JTextField txtCilindraje  - JTextField txtAvaluoComercial  - JTextField txtTipoUso  - JButton btnCalcular  - JLabel lblResultado  - Controlador controlador | + getMarca(): String  + getModelo(): String  + getAnoFabricacion(): int  + getCilindraje(): int  + getAvaluoComercial(): double  + getTipoUso(): String  + setResultado(impuesto: double): void  + getBtnCalcular(): JButton |
| Controlador |  |  |
| CalculadoraImpuestoss |  |  |

**Deploy de Desarrollo**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Como podemos ver en este caso, se crea una aplicación en java usando NetBeans la cual tiene como nombre de proyecto ‘*CalculadoraImpuestoss*’, la cual es la responsable de tener dentro 5 clases las cuales son:  
  
**CalculadoraImpuestos.java**  
  
Es la encargada de contener la lógica del cálculo de los avalúos de impuestos, cada uno de estos valores se pueden modificar ingresado al archivo fuente.

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

**CalculadoraImpuestosApp.java**Es la encargada de ejecutar la vista del proyecto es la base inicial del código la cual despliega las demás clases, por lo tanto es la mas importante.

**Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.  
  
Controlador.java**

El controlador es el encargado de realizar la creación de objetos, calcula impuestos añade en la vista la información de cada uno de los parámetros.

**Imagen que contiene Tabla

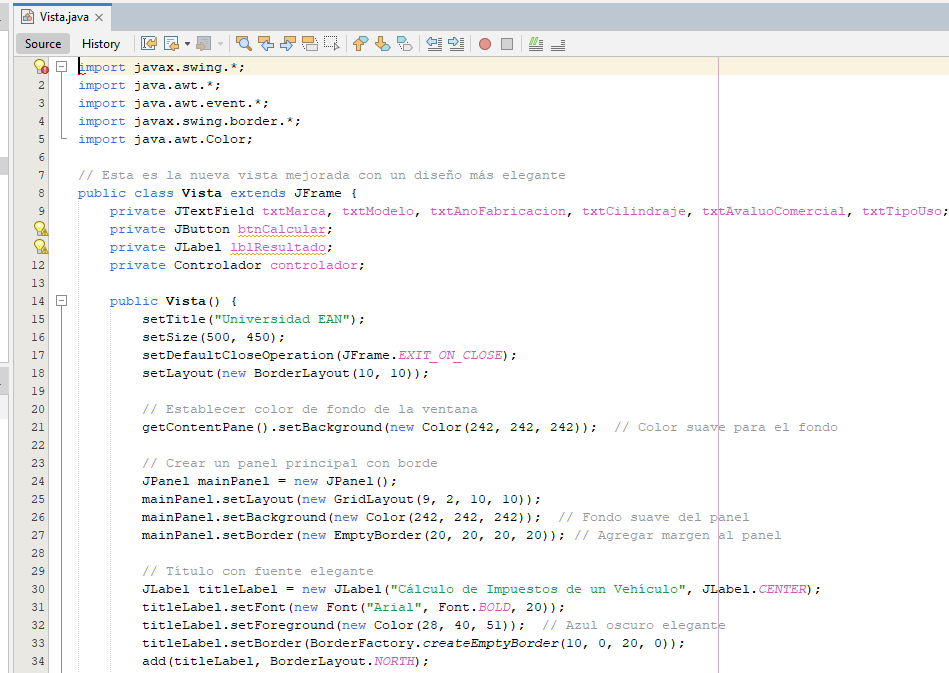
El contenido generado por IA puede ser incorrecto.**

**Vehiculo.java**

La clase Vehiculo.java representa el modelo de la aplicación y tiene la función de almacenar y gestionar la información básica de un vehículo. Esta clase no tiene lógica compleja, sino que se dedica a ser un contenedor de datos que puede ser utilizado por otras partes del sistema, como la clase CalculadoraImpuestos, para calcular el impuesto sobre el vehículo en función de sus atributos.

**Interfaz de usuario gráfica, Texto, Aplicación

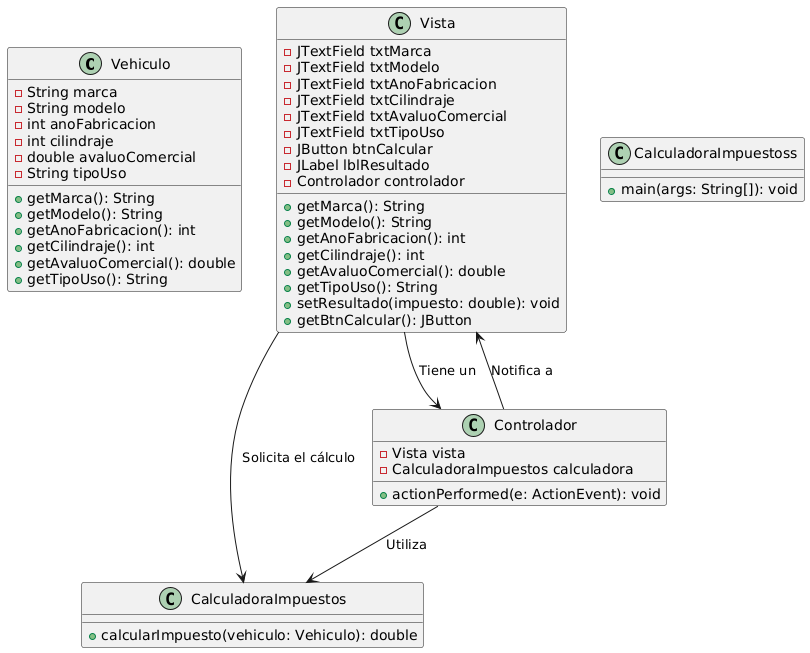
El contenido generado por IA puede ser incorrecto.**

**Vista.Java**La clase Vista.java es la interfaz gráfica (GUI) de la aplicación. Su objetivo principal es permitir al usuario interactuar con el sistema de manera visual. Es la parte del programa que el usuario ve y con la que interactúa directamente para ingresar los datos del vehículo y obtener el resultado del cálculo del impuesto.  
  
****

**Ejecución de app**

La ejecución de la aplicación se realiza mediante el llamado a CalculadoraImpuestosApp.java, la cual despliega la GUI para la interactividad de los usuarios.  
  
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Diagrama de clases uml**

LINK GITHUB: https://github.com/mehirakiva/EAN\_Patrones\_de\_diseno\_software/tree/77fa7a7e5c3732dc5d15cfeae7498cb553f6a4cf/Actividad2

**CONCLUSIONES**

El uso del patrón de diseño Modelo-Vista-Controlador (MVC) ha sido una excelente decisión, ya que permitió separar la lógica de negocio (cálculo de impuestos) de la interfaz de usuario. Esto facilita la gestión de la aplicación, haciendo que sea más fácil de mantener y extender en el futuro

El proyecto ha alcanzado sus objetivos principales, proporcionando una herramienta útil para calcular impuestos sobre vehículos de manera efectiva y eficiente. Este tipo de aplicaciones facilita la vida de los usuarios y mejora la precisión en la gestión de impuestos, además de permitir futuras ampliaciones y adaptaciones.