Nombre de la Materia
Desarrollo de Software

Nombres de los Alumnos
Paula Alejandra Casallas Bonilla

Hermann Vallejo Rodríguez

Estefania Agudelo Bustamante

Nombre del Profesor
Dilsa Enith Triana

Grupo y Guía
Guía 2

Fecha
03/03/2025

Introducción

El sistema de Cálculo de Impuestos de Vehículos es una aplicación desarrollada en Java utilizando el patrón de diseño Modelo-Vista-Controlador (MVC). Su propósito es proporcionar a los usuarios una herramienta fácil de usar para calcular el impuesto vehicular de manera automática, teniendo en cuenta factores como el avalúo comercial, el año de fabricación y el tipo de uso del vehículo.

La aplicación cuenta con una interfaz gráfica intuitiva, desarrollada con Swing, que permite a los usuarios ingresar los datos requeridos y obtener el valor del impuesto de forma inmediata. El sistema también garantiza un cálculo preciso y una presentación clara de los resultados, evitando la notación científica en los valores monetarios.

Objetivos

- Automatizar el cálculo de impuestos de vehículos mediante una lógica clara y precisa basada en reglas establecidas.
- Implementar el patrón MVC para garantizar una estructura organizada, separando la lógica de negocio, la interfaz de usuario y el control de eventos.
- Proporcionar una interfaz gráfica intuitiva que permita a los usuarios ingresar los datos del vehículo y visualizar el impuesto sin complicaciones.
- Evitar la notación científica en los valores monetarios, asegurando una presentación adecuada del impuesto calculado.
- Facilitar la escalabilidad del sistema, permitiendo futuras mejoras como la actualización dinámica de tarifas o la integración con bases de datos.

Desarrollo

Actividad 2. Java e interfaces gráficas

1. Lea detenidamente el capítulo 5 del libro de Villalobos & Casallas (2006), así como el capítulo 12 de Deitel & Deitel (2016).

2. Conforme grupos de máximo 3 estudiantes entre sus compañeros de bloque e indique al tutor quiénes son los miembros de cada grupo.

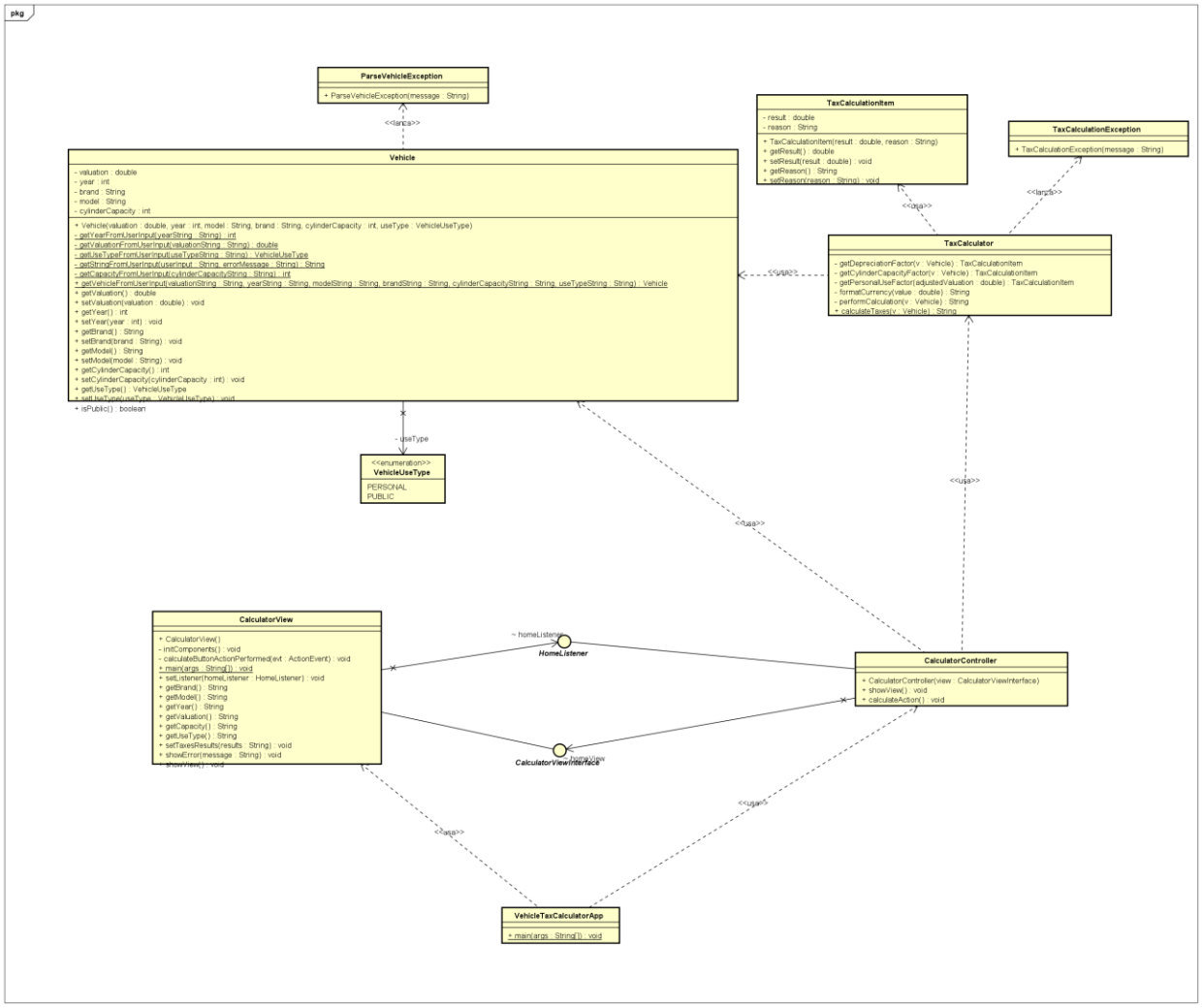
3. Revise el objeto de aprendizaje "***Instalación y organización de grupos en GitHub***".

4. Lea el enunciado de los problemas que hace parte del proyecto «Cálculo de impuestos de un carro» También estudie los requerimientos funcionales y el modelo de clases del proyecto.

5. Una vez realizadas todas las tareas, suba el proyecto entero a un repositorio distribuido de alguno de los miembros del grupo. Este repositorio puede ser en GitHub o GitLab y comparta la dirección con el tutor.

6. Finalmente, elabore un informe técnico donde describa las decisiones realizadas en el código realizado, así como la dirección del repositorio donde quedó almacenado la versión final del proyecto. Suba el informe al aula virtual. Sólo un miembro del grupo debe realizar la entrega a través del aula.

Diagrama Impuesto Vehicular



- DESCRIPCION:

Arquitectura MVC (Modelo-Vista-Controlador)

- **Modelo:** Representa los datos y la lógica de negocio. Los tipos de datos que forman parte de esta capa son: las clases **Vehicle** que se utiliza para representar los datos de un vehículo, así como también para realizar validaciones sobre estos valores, la clase Vehicle lanza una excepción **ParseVehicleException** cuando recibe valores incorrectos para sus campos, **TaxCalculator** que se encarga de realizar el cálculo de impuesto a partir de una instancia de Vehicle, esta clase usa a la clase **TaxCalculationItem** para devolver resultados intermedios en el cálculo del impuesto junto a una cadena descriptiva sobre cómo se realizó el cálculo, y finalmente, la enumeración **VehicleUseType** se utiliza para identificar el tipo de uso del vehículo (particular o publico).

- **Vista:** Representa la interfaz de usuario. La clase **CalculatorView** es la vista que permite al usuario ingresar los datos del vehículo y ver los resultados del cálculo de impuestos, utiliza métodos getter para devolver al controlador los datos ingresados por el usuario y una interfaz **HomeListener** para notificar cuando se pulsa el botón calcular. Además provee un método setter para establecer el resultado del cálculo del impuesto.
  Esta vista utiliza un **GridBagLayout** para organizar los componentes de interfaz gráfica.

- **Controlador:** Actúa como intermediario entre la vista y el modelo. La clase **CalculatorController** maneja la lógica de la aplicación, recibe los datos de la vista y actualiza el modelo con los datos y las acciones del usuario.

Uso de Enumeraciones

- **VehicleUseType:** Es una enumeración que define los tipos de uso del vehículo (PERSONAL y PUBLIC). Esto permite un manejo más seguro y legible de los tipos de uso, evitando errores comunes como cadenas de texto mal escritas.

Manejo de la Interfaz de Usuario

- **ComponentView:** Utilizamos componentes de Swing para crear una interfaz gráfica que permite al usuario ingresar los datos del vehículo. Los componentes incluyen campos de texto, un botón para calcular el impuesto y un área de texto para mostrar los resultados.

- **JSpinner:** Se utiliza para ingresar el año del vehículo, asegurando que el usuario solo pueda ingresar valores numéricos válidos.

- **JComboBox:** Se utiliza para seleccionar el tipo de uso del vehículo (particular o público), lo que facilita la selección y evita errores de entrada.

- **GridBagLayout**. Layout Swing que permite componer interfaces complejas en un único JPanel o JFrame manejando posiciones de cuadricula, tamaños relativos a cuadricula y pesos.

Lógica de Cálculo de Impuestos

- **TaxCalculator:** Esta clase implementa la lógica de cálculo de impuestos y recibe una instancia de la clase Vehicle actualizada con los valores ingresados por el usuario. La clase expone un método calculateTaxes() que retorna una cadena descriptiva del cálculo realizado.

Cálculo del impuesto:

Se define un factor de depreciación de la siguiente manera:

- Vehículo con antigüedad menor a 3 años: sin depreciación.
- Vehículo con antigüedad entre 3 y 10 años: valor se reduce al 85% del valor original.
- Vehículo con antigüedad entre 10 y 20 años: valor se reduce al 70% del valor original.
- Vehículo con antigüedad mayor a 20 años: valor se reduce al 50% del valor original.

El cálculo también incluye un recargo por cilindraje:

- Vehículo con cilindraje menor o igual a 1000cc: sin recargo.
- Vehículo con cilindraje entre 1000cc y 2000cc: 10% de recargo.
- Vehículo con cilindraje mayor a 2000cc: 20% de recargo.

Tarifa de tipo de uso:

- Vehículos particulares con avaluó menor a $55.679.000: 1.7%
- Vehículos particulares con avaluó entre $55.679.000 y $125.274.000: 2.7%
- Vehículos particulares con avaluó mayor a $125.274.000: 3.7%
- Vehículos de transporte público: 0.7%

Calculo final:

$$impuesto = avaluo\ ajustado\ a\ depreciacion * tarifa\ de\ tipo\ de\ uso * recargo\ de\ cilindraje$$

- **Formato de Salida:** El resultado del cálculo se formatea utilizando **DecimalFormat** para mostrar el impuesto con dos decimales y separadores de miles, lo que mejora la legibilidad del resultado.

Manejo de Eventos

- **HomeListener:** Es una interfaz que define un método calculateAction(), que es implementado por **CalculatorController**. Cuando el usuario hace clic en el botón "Calcular Impuesto", la vista notifica al controlador a través de este método, y el controlador ejecuta la lógica de cálculo y actualiza la vista con los resultados.

Validación y Conversión de Datos

- **Conversión de Datos:** La clase Vehicle se encarga de generar una instancia basada en las entradas del usuario como cadenas de texto en los TextFields de la vista si los datos son válidos o lazar una excepción si hay valores incorrectos, por ejemplo, un avaluó negativo.

- **Manejo de Errores:** La clase Vehicle maneja la validación de sus campos lanzando una excepción que genera un mensaje en pantalla con el error.

Inicialización de la Aplicación

- **VehicleTaxCalculatorApp:** Es la clase principal que inicia la aplicación. Crea una instancia de CalculatorController, CalculatorView y muestra la vista principal.

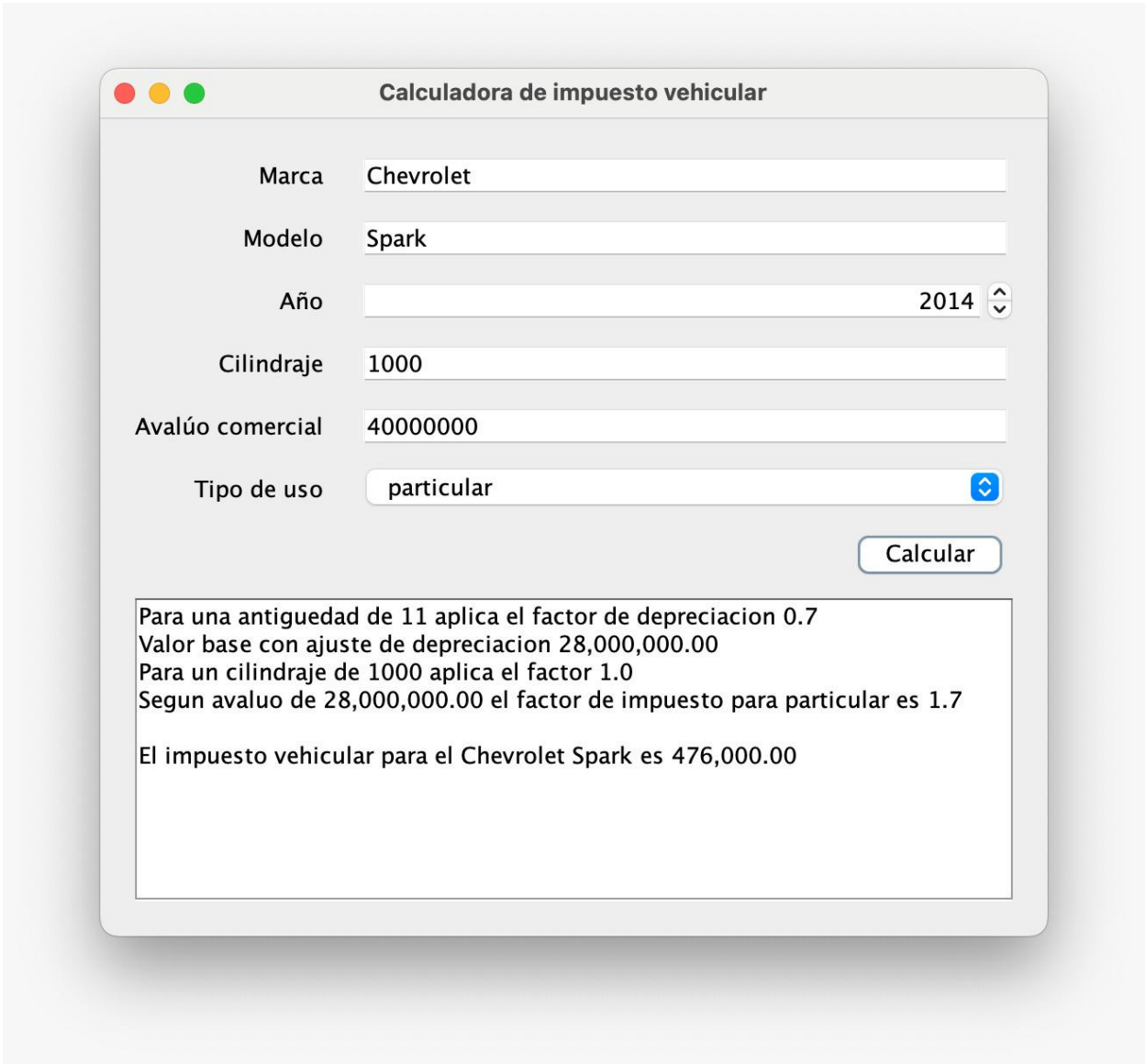
Reusabilidad y Extensibilidad

- **TaxCalculator:** La lógica de cálculo de impuestos está encapsulada en una clase separada, lo que facilita su reutilización en otras partes de la aplicación o en futuras actualizaciones.

- **Vehicle:** La clase Vehicle es un modelo de datos que puede ser extendido o modificado para incluir más atributos o comportamientos en el futuro.

Interfaz Gráfica Responsiva

- **CalculatorView:** La interfaz gráfica está diseñada para ser responsiva, con etiquetas y campos de texto alineados adecuadamente. El uso de JScrollPane para el área de resultados asegura que el texto largo no desborde la interfaz.



Conclusiones

- El trabajo realizado es un buen ejemplo de cómo implementar una aplicación simple utilizando el patrón MVC en Java. Las decisiones tomadas en el diseño permiten una buena separación de preocupaciones, lo que facilita la comprensión y el mantenimiento del código. Sin embargo, hay áreas donde se puede mejorar, como la validación de entradas y el manejo de excepciones.

- La construcción de una interfaz gráfica efectiva es crucial para el éxito de una aplicación. Un buen diseño no solo mejora la experiencia del usuario, sino que también puede aumentar la productividad y satisfacción general.

Referencias

Villalobos S., J. A., & Casallas, R. (2006). Fundamentos de programación: aprendizaje activobasado en casos: un enfoque moderno usando Java, UML, Objetos y Eclipse. Capítulo 5. https://www-ebooks7-24-com.bdbiblioteca.universidadean.edu.co/stage.aspx?il=&pg=&ed=