



**Universidad de las Fuerzas Armadas - ESPE**

**Pregrado SII OCT24 – MAR25**

**Programación Orientada a Objetos – Nrc: 1323**

**Control de Lectura 2: Creación de Objetos y UML**

**Nombre de los integrantes:**

Daniel Alejandro Quiguango Delgado

**Nombre del docente:**

Mgtr. Luis Enrique Jaramillo Montaña

**Fecha de entrega:** 08 de diciembre del 2024

## Contenido

Introducción.....	4
1. Objetivos.....	4
1.1. Objetivos General.....	4
1.2. Objetivos Específicos .....	4
2. Marco teórico .....	4
Componentes.....	5
3. Resultados Diagramas de clase UML.....	6
4. Resumen de resolución.....	8
Análisis de las relaciones.....	8
Resumen de Funcionalidades.....	8
5. Código en JAVA Apache NetBeans.....	9
Interfaces.....	9
Clases con Implementación de Interfaces .....	10
6. Conclusiones.....	15
7. Recomendaciones.....	15
8. Referencias Bibliográficas .....	16
9. Apéndice.....	16

## Tabla de figuras

<b>Figura 1.....</b>	<b>6</b>
<b>Figura 2.....</b>	<b>7</b>
<b>Figura 3.....</b>	<b>9</b>
<b>Figura 4.....</b>	<b>9</b>
<b>Figura 5.....</b>	<b>10</b>
<b>Figura 6.....</b>	<b>11</b>
<b>Figura 7.....</b>	<b>11</b>
<b>Figura 8.....</b>	<b>12</b>
<b>Figura 9.....</b>	<b>12</b>
<b>Figura 10.....</b>	<b>13</b>
<b>Figura 11.....</b>	<b>13</b>
<b>Figura 12.....</b>	<b>14</b>
<b>Figura 13.....</b>	<b>14</b>
<b>Figura 14.....</b>	<b>15</b>

## Introducción

En este informe se presenta el desarrollo e implementación de un sistema de gestión de talleres de vehículos utilizando la programación orientada a objetos (POO). El objetivo principal es diseñar un sistema que permita registrar talleres, gestionar vehículos, registrar ventas y consultar información clave de manera eficiente. El sistema es modular, escalable y puede ser aplicado a la gestión de talleres de diferentes tipos.

## 1. Objetivos

### 1.1. Objetivos General

- Crear un sistema de gestión para talleres automotrices que permita el registro de los talleres, la administración de vehículos, el control de ventas y la consulta de datos importantes para mejorar la eficiencia operativa.

### 1.2. Objetivos Específicos

- Desarrollar un sistema utilizando programación orientada a objetos o POO que posibilite el registro de talleres y los vehículos relacionados.
- Desarrollar funcionalidades que permitan registrar y consultare las ventas efectuadas por cada taller.
- Ofrecer herramientas que faciliten la visualización del historial de ventas y la actualización de resultados de forma eficiente.

## 2. Marco teórico

La programación orientada a objetos (POO) es un paradigma de programación que permite organizar el software en torno a objetos que interactúan entre sí. Este enfoque facilita la reutilización de código, el modularidad y el mantenimiento del sistema (Meyer, 1997).

Conceptos clave utilizados en este proyecto incluyen:

- **Clases y Objetos:** Representan entidades del sistema (e.g., Taller, Vehículo) y se definen como la base de la programación orientada a objetos. Las clases contienen atributos

y métodos que describen el comportamiento de los objetos que se crean a partir de ellas (Keene, 1989).

- **Encapsulación:** Protege los atributos internos de las clases mediante métodos públicos, permitiendo que las modificaciones internas de una clase no afecten el resto del sistema. Esto es clave para el mantenimiento y escalabilidad del código (Universitat Oberta de Catalunya, s.f.).
- **Composición:** Relación entre clases donde una clase contiene instancias de otra, indicando una relación de "todo-parte". Por ejemplo, un taller tiene una lista de vehículos; esta relación asegura que los vehículos no puedan existir sin el taller (Cachero Castro, 2013).
- **Agregación:** Relación más flexible entre clases, como la relación entre talleres y ventas, donde un objeto puede existir independientemente del otro. Este enfoque permite un diseño más modular y reutilizable (Fontela & Paez, 2015).

La implementación de estos conceptos en el sistema presentado permite estructurar las relaciones entre talleres, vehículos y ventas, facilitando la gestión eficiente y ampliando las posibilidades de personalización y escalabilidad.

## Componentes

Un diagrama de clases cuenta con: el nombre, los atributos, las operaciones o comportamientos o métodos y las relaciones, elementos que se describen brevemente a continuación.

- **Nombre:** sirve de identificador para la clase, cada una de estas debe tener un nombre que no se repita
- **Atributos:** son las propiedades y características que tienen las clases, se los representa con el nombre del objeto, seguido del tipo de dato que es el atributo.
- **Operaciones:** también conocidos como métodos en Programación Orientada a Objetos, son las acciones que pueden ser realizadas por las instancias de esa clase y que

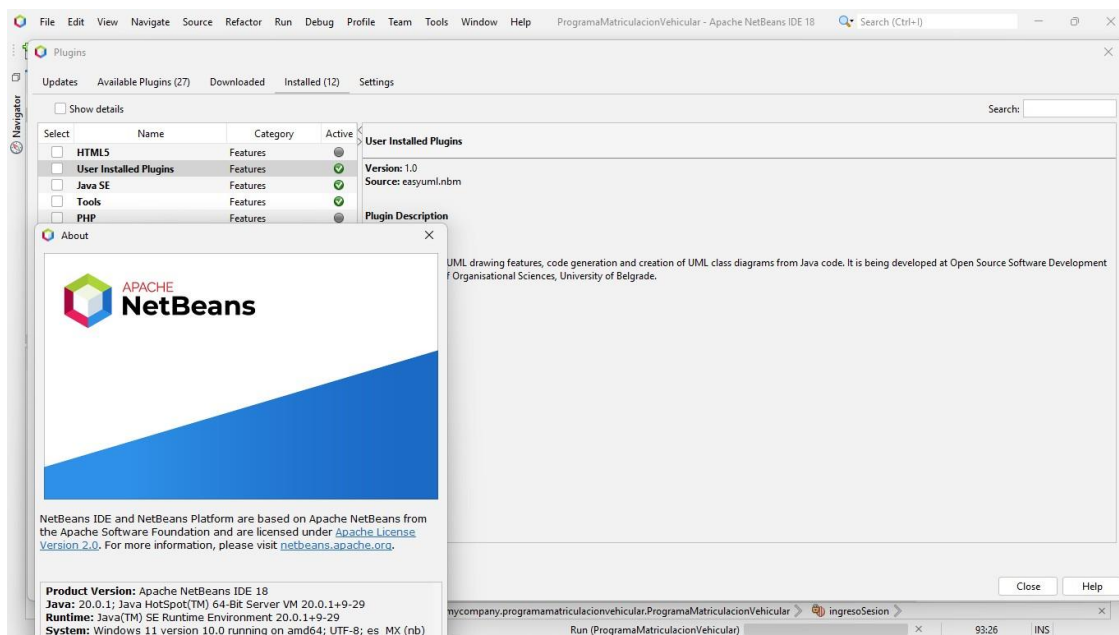
representa el comportamiento que tendrá dicha clase, se representan con: el nombre de la operación o método, los parámetros que tendrá y el tipo de retorno.

### 3. Resultados Diagramas de clase UML

A continuación, podemos observar en la Figura 1. el plug-in instalado para crear diagramas UML en NetBeans.

**Figura 1.**

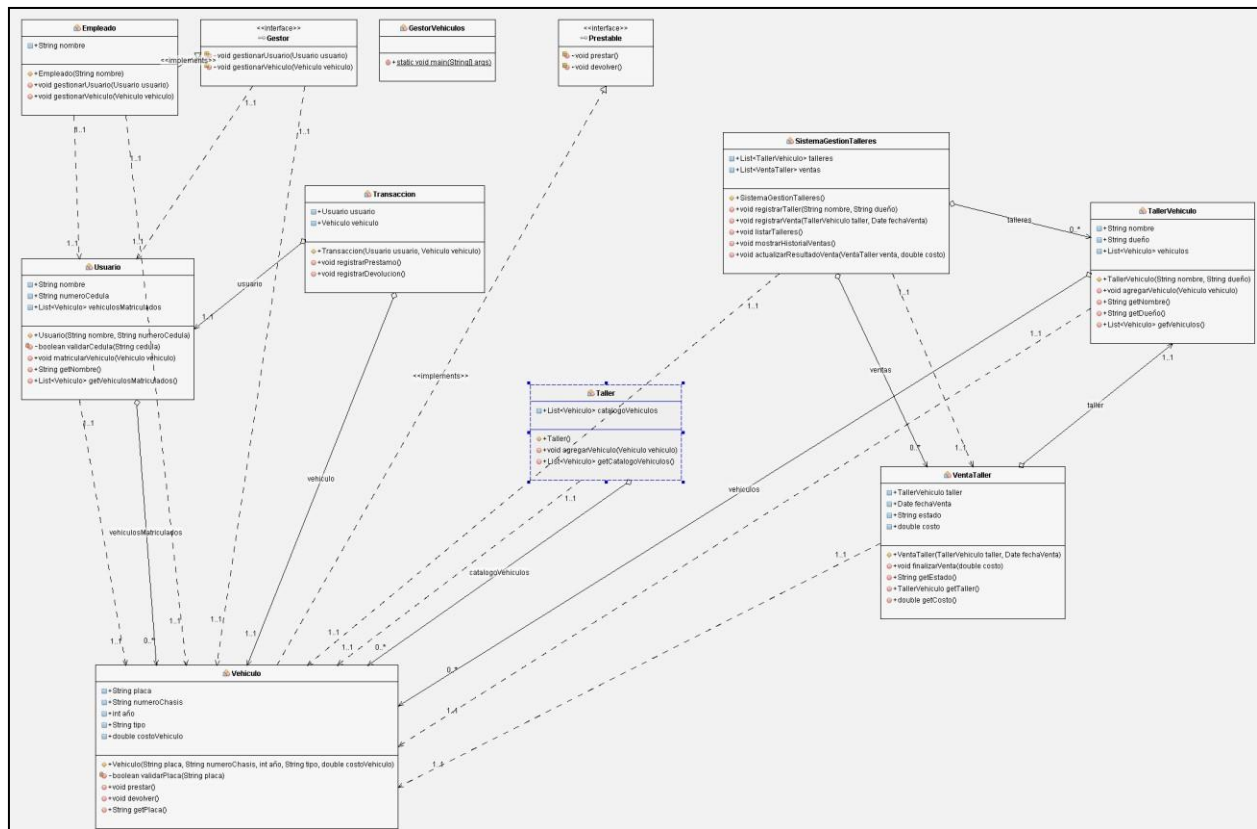
Plugin instalado easyUML instalado en Windows 11



Posteriormente, podemos observar en la Figura 2. la creación de un ejemplo de un diagrama de clases empleando el plug-in easyUML llamado GestorVehiculos.

**Figura 2.**

Diagrama de clase GestorVehiculos UML



#### 4. Resumen de resolución

- **Vehículo:** Representa los vehículos disponibles en un Taller, con atributos como placa, numero de chasis, año, evaluo, tipo, costo de matrícula, y métodos como prestar() y devolver()).
- **Usuario:** Los usuarios pueden tener una lista de vehículos matriculados. Se relaciona con Vehículo mediante composición.
- **Taller:** Administra un catálogo de vehículos y se relaciona con Vehículo por asociación.
- **Empleado:** Gestiona usuarios y Vehículos, y se relaciona con Usuario y Vehículo por asociación.
- **Transacción:** Registra préstamos y devoluciones. Se relaciona con Usuario y Vehículo por agregación.

#### Análisis de las relaciones

- **Composición:** Usuario tiene una lista de Vehiculo. Si se elimina un Usuario, se eliminan sus vehículos matriculados.
- **Agregación:** Transaccion tiene referencias a Usuario y Vehiculo. Las transacciones pueden existir sin eliminar los usuarios o vehículos.
- **Asociación:** Taller tiene una lista de Vehiculo. Los vehículos pueden existir independientemente del taller.

#### Resumen de Funcionalidades

- **Registrar Taller de Vehículo:** Se puede registrar un taller con un nombre y un dueño, y agregar vehículos a ese taller.
- **Crear Ventas de Talleres:** Se registra una venta para un taller con la fecha de la venta.
- **Consultar Información:**
  - Listar todos los talleres registrados con sus respectivos vehículos.



- Mostrar el historial de ventas de talleres de vehículos, incluyendo los talleres y los resultados.
- **Actualizar Resultados de la Venta:** Cambiar el estado de una venta de "Pendiente" a "Finalizado", añadiendo el costo.

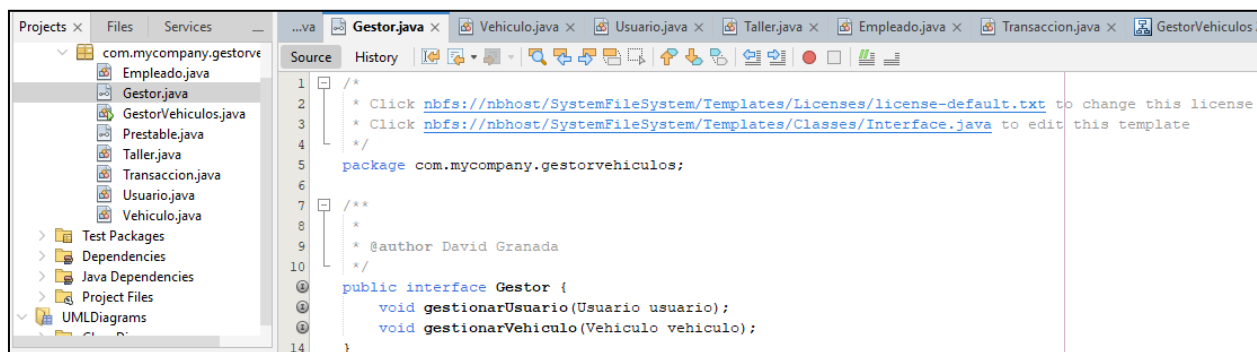
## 5. Código en JAVA Apache NetBeans

### Interfaces

Primero, definamos algunas interfaces que representen las acciones que pueden realizar las clases como se puede observar en las siguientes figuras.

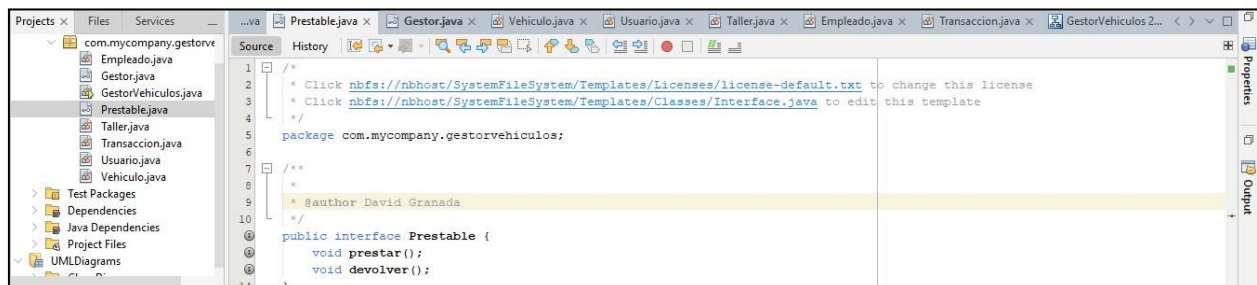
**Figura 3.**

Creación de la Interfaz Prestable



**Figura 4.**

Creación de la Interfaz Gestor

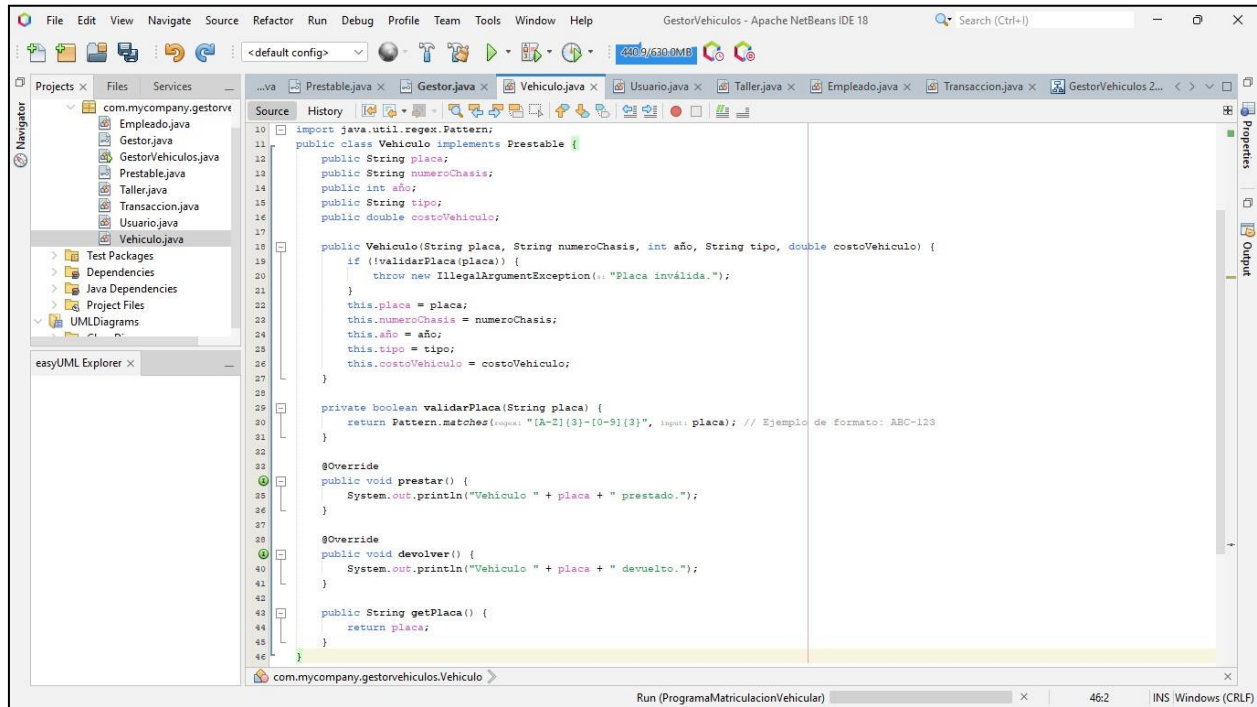


## Clases con Implementación de Interfaces

Ahora, adaptaremos las clases para que implementen estas interfaces como se puede observar en las Figuras 10 a la 16.

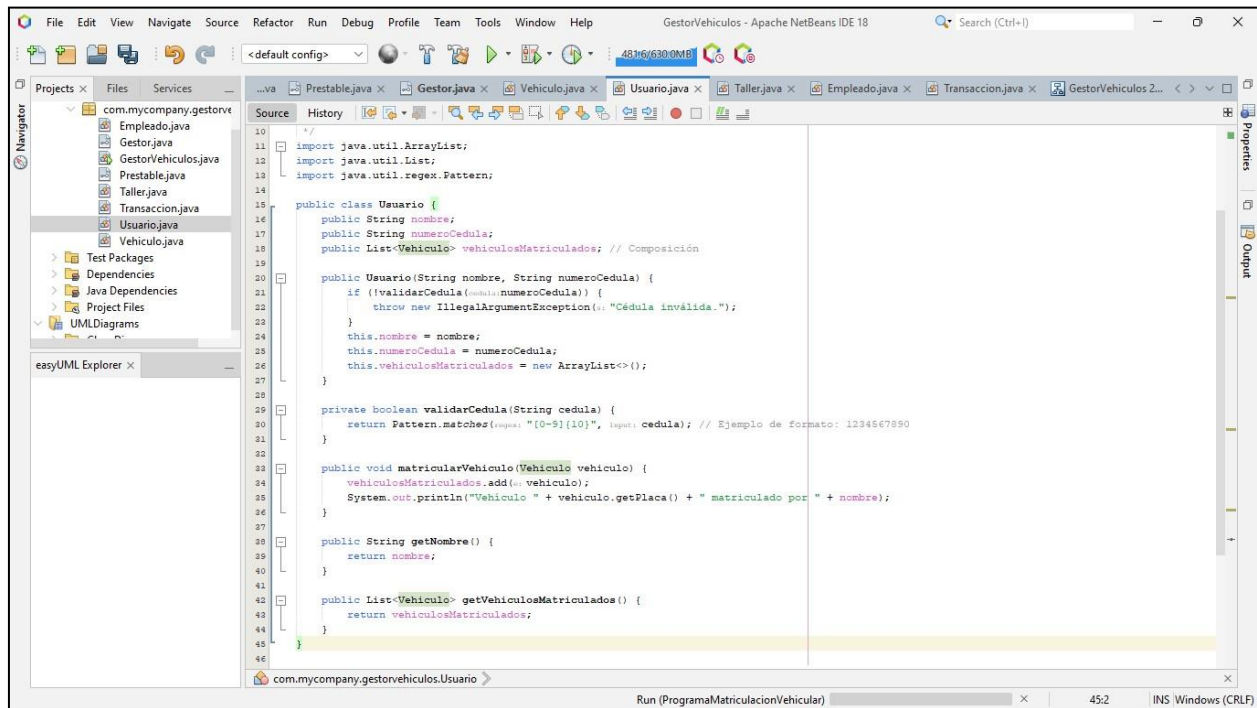
**Figura 5.**

### Creación Clase Vehiculo



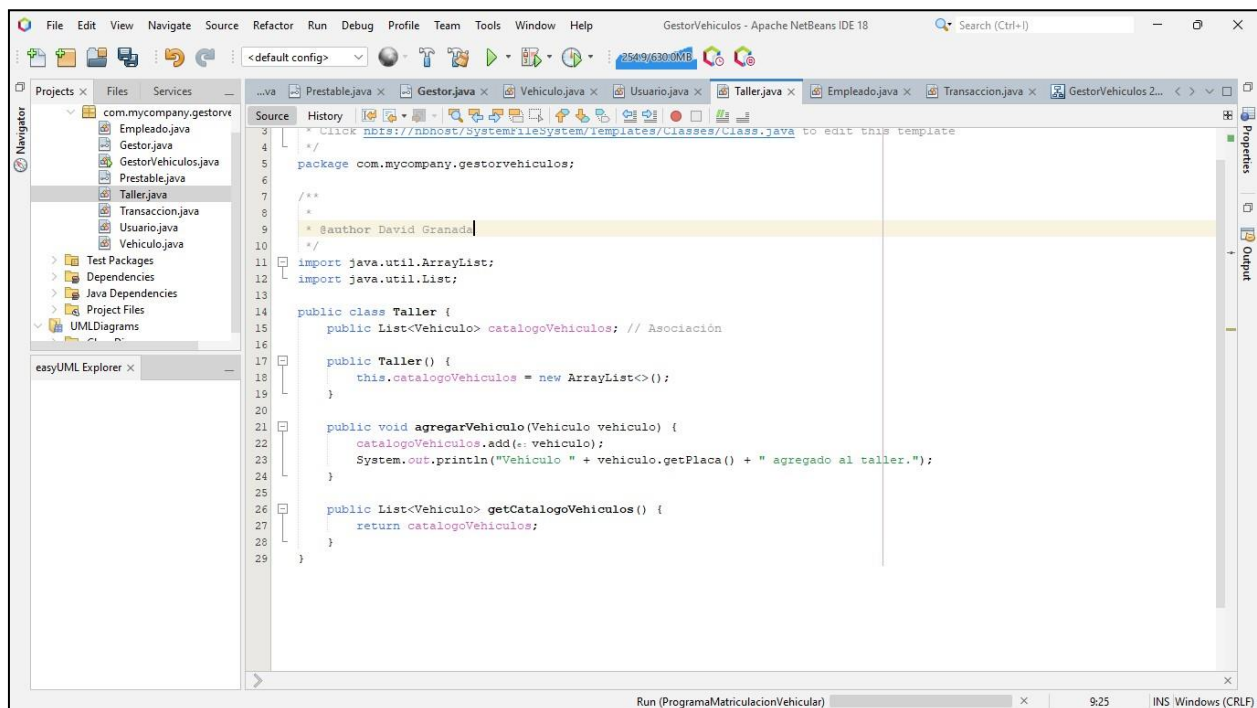
**Figura 6.**

## Creación Clase Usuario (Composición)



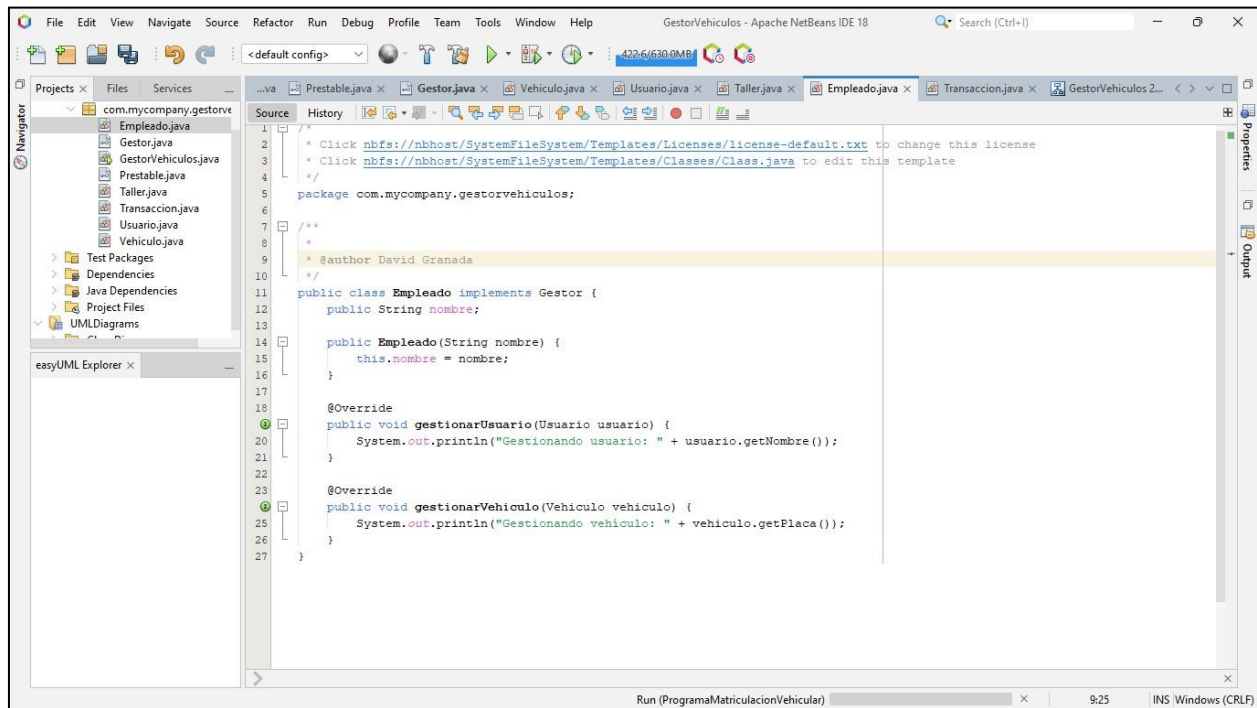
**Figura 7.**

## Creación Clase Taller (Asociación)



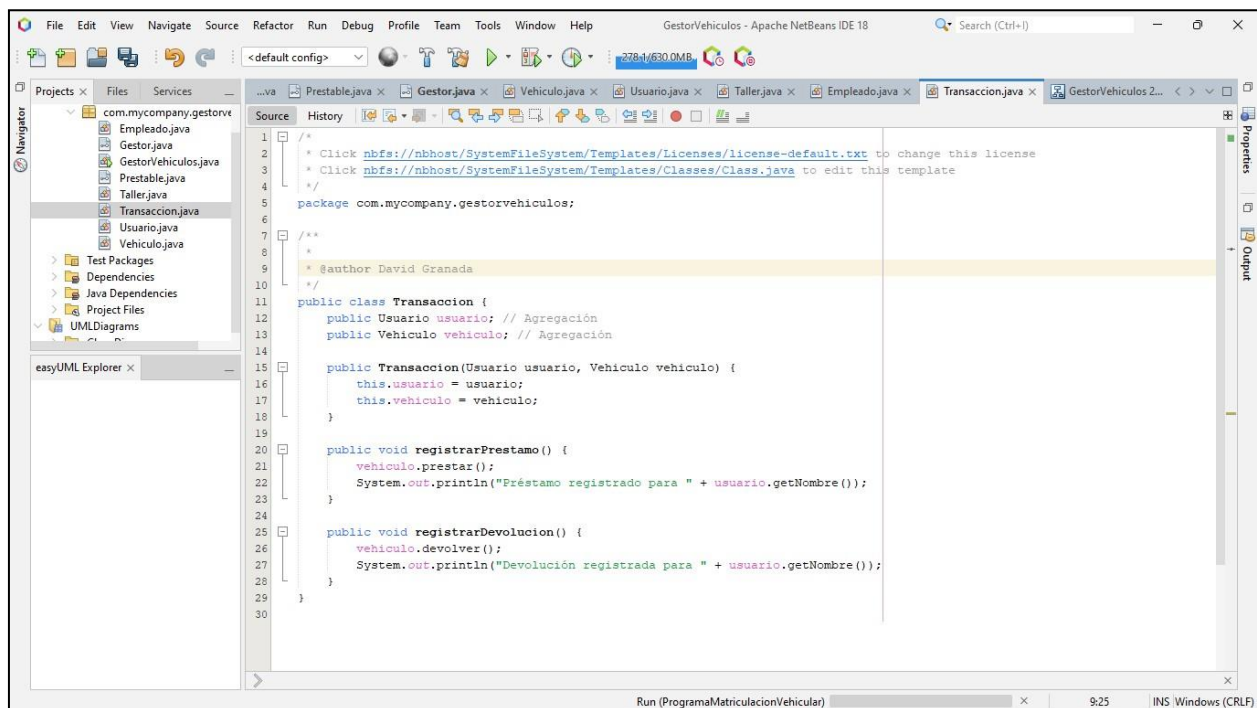
**Figura 8.**

## Creación Clase Empleado (Asociación)



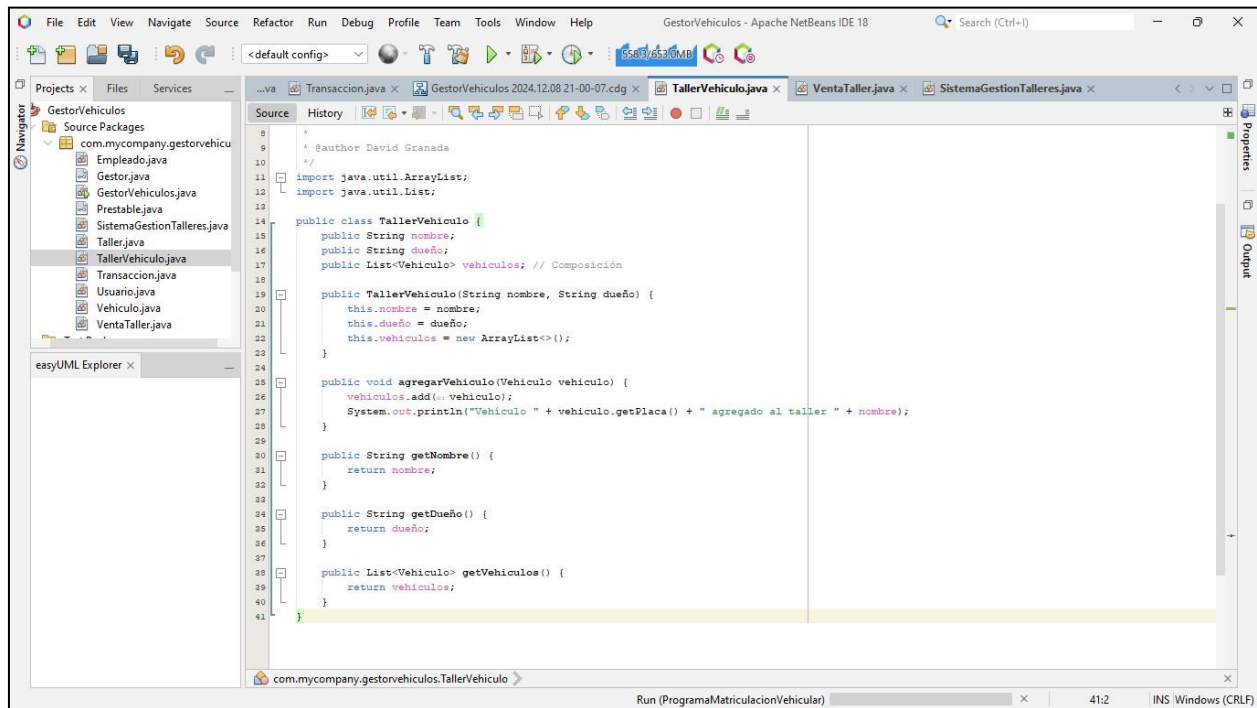
**Figura 9.**

## Clase Transaccion (Agregación)



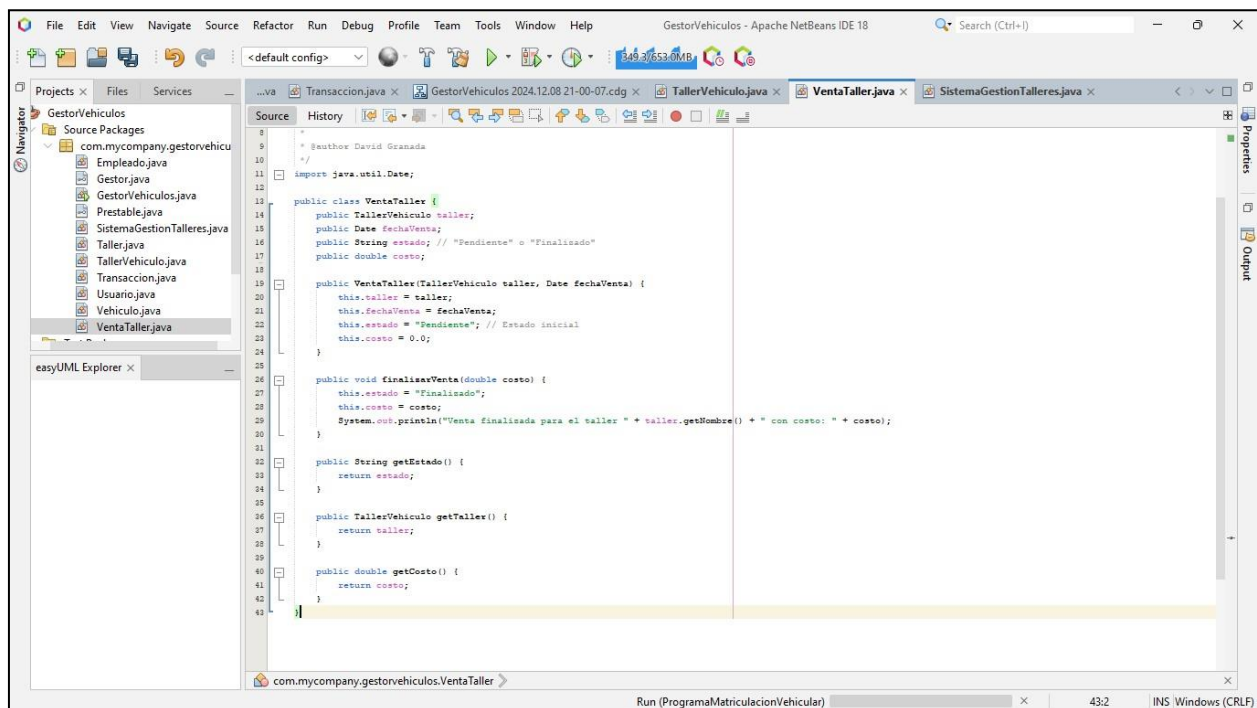
**Figura 10.**

## Clase Taller Vehiculo



**Figura 11.**

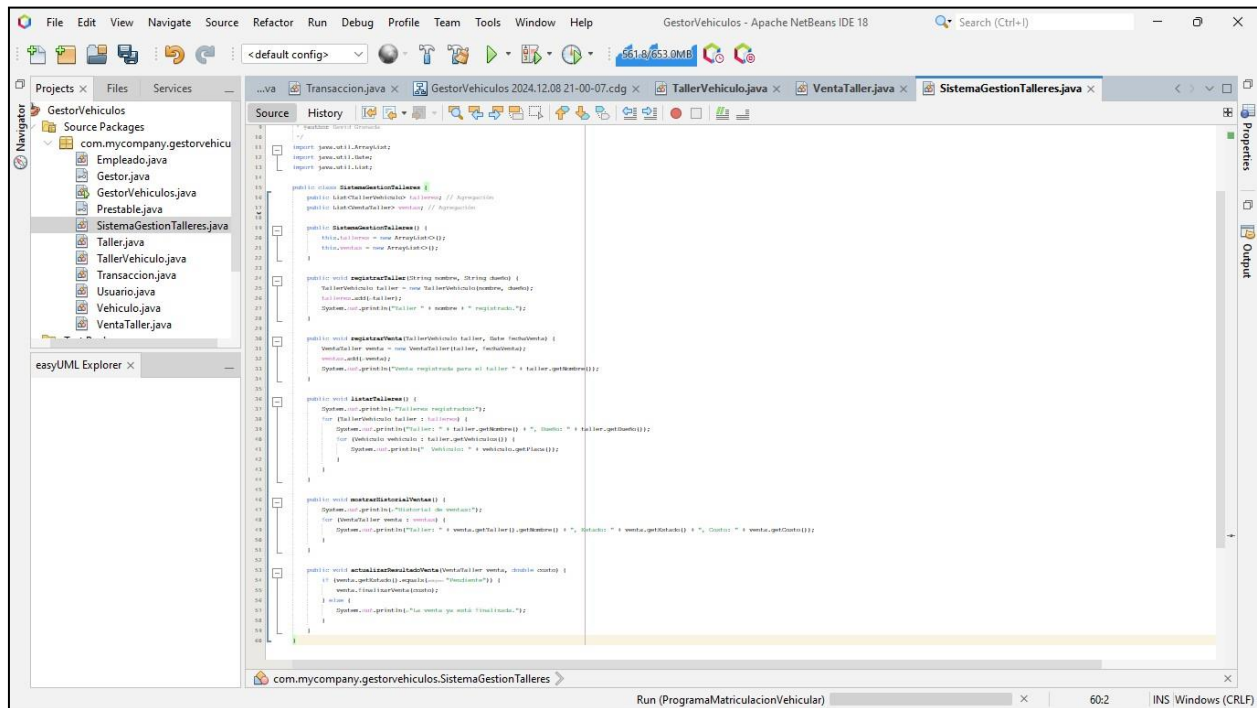
## Clase Venta Taller





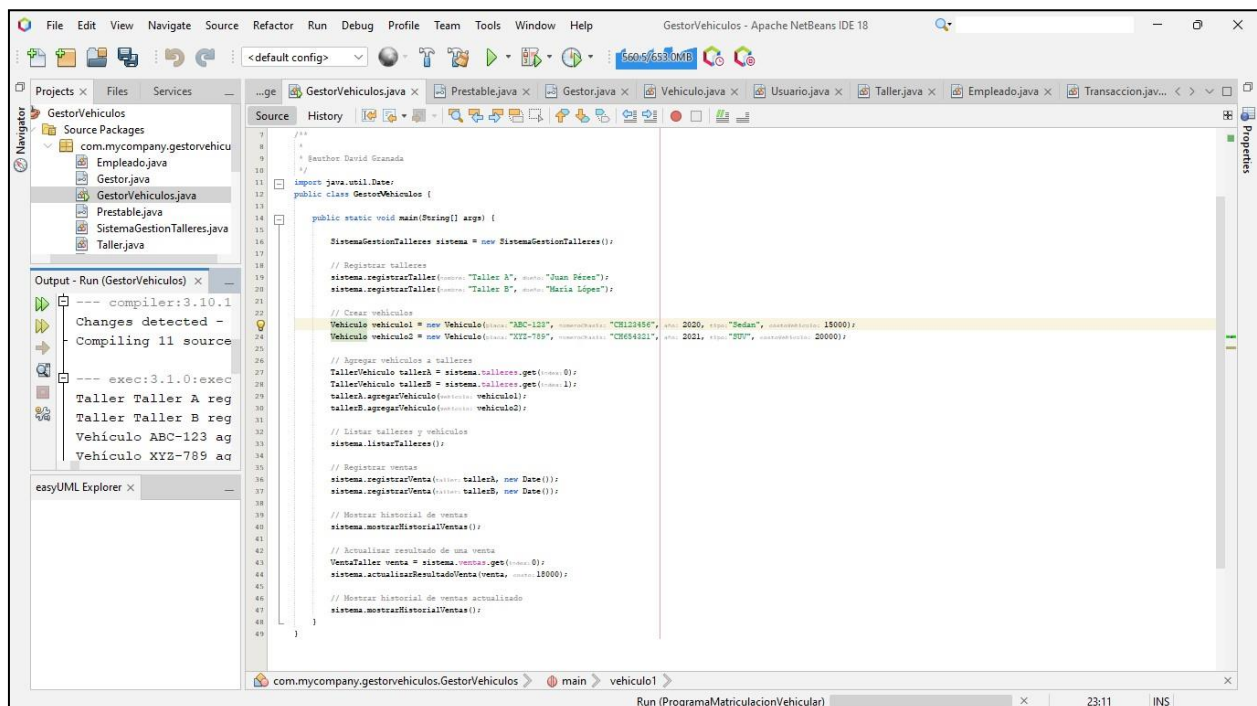
**Figura 12.**

## Clase SistemaGestionTalleres



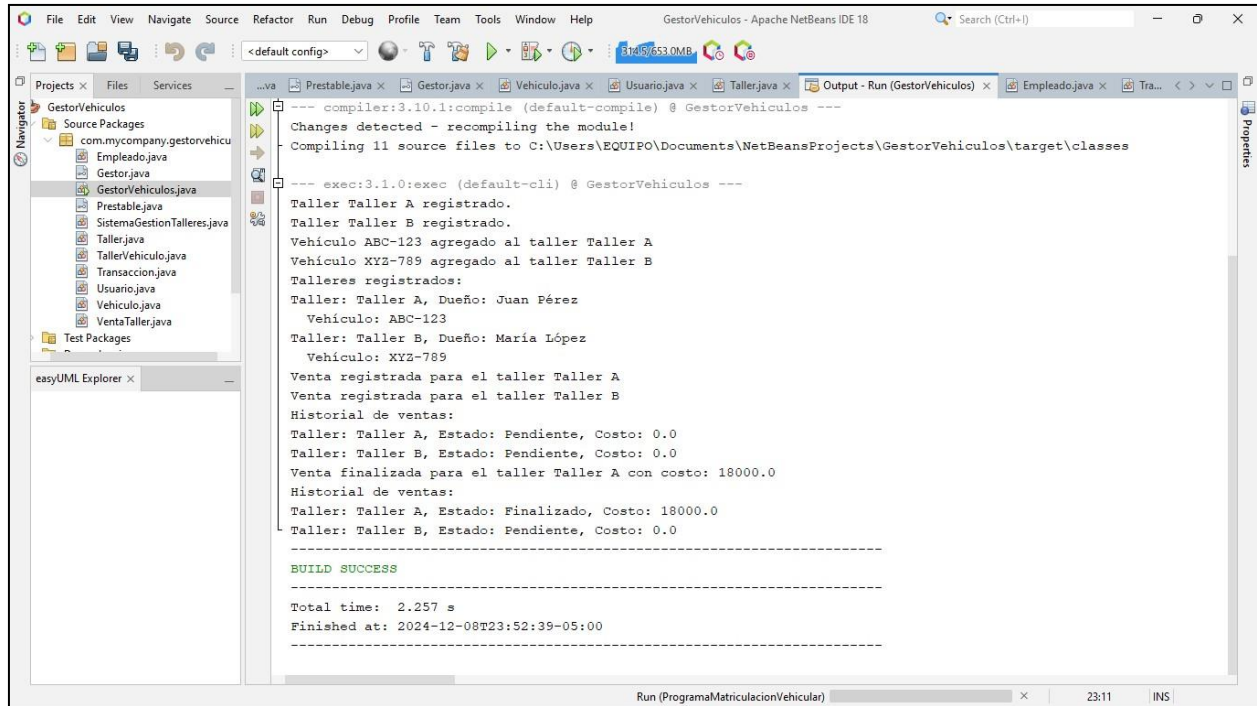
**Figura 13.**

## Creación Clase Principal GestorVehicular



**Figura 14.**

Salida del programa por consola



```

--- compiler:3.10.1:compile (default-compile) @ GestorVehiculos ---
Changes detected - recompiling the module!
Compiling 11 source files to C:\Users\EQUIPO\Documents\NetBeansProjects\GestorVehiculos\target\classes

--- exec:3.1.0:exec (default-cli) @ GestorVehiculos ---
Taller Taller A registrado.
Taller Taller B registrado.
Vehiculo ABC-123 agregado al taller Taller A
Vehiculo XYZ-789 agregado al taller Taller B
Talleres registrados:
Taller: Taller A, Dueño: Juan Pérez
    Vehiculo: ABC-123
Taller: Taller B, Dueño: María López
    Vehiculo: XYZ-789
Venta registrada para el taller Taller A
Venta registrada para el taller Taller B
Historial de ventas:
Taller: Taller A, Estado: Pendiente, Costo: 0.0
Taller: Taller B, Estado: Pendiente, Costo: 0.0
Venta finalizada para el taller Taller A con costo: 18000.0
Historial de ventas:
Taller: Taller A, Estado: Finalizado, Costo: 18000.0
Taller: Taller B, Estado: Pendiente, Costo: 0.0

BUILD SUCCESS

Total time: 2.257 s
Finished at: 2024-12-08T23:52:39-05:00
  
```

## 6. Conclusiones

- El sistema implementado alcanza los objetivos establecidos al facilitar la gestión efectiva de talleres, vehículos y ventas.
- El Uso de POO favoreció el modularidad y escalabilidad del sistema, lo que permite futuras ampliaciones.
- El diseño UML facilitó la visualización de las relaciones entre las clases y oriento el proceso de implementación

## 7. Recomendaciones

- Integrar una base de datos para almacenar de manera permanente la información de talleres, vehículos y ventas.
- Desarrollar una interfaz grafica que facilite la interacción del usuario con el sistema.
- Expandir las funcionalidades para incluir reportes detallados y la gestión de clientes.

## 8. Referencias Bibliográficas

- Cachero Castro, C. (2013). Introducción a la programación orientada a objetos. Textos docentes. Recuperado de [https://sedici.unlp.edu.ar/bitstream/handle/10915/29797/Clase\\_2013.pdf?sequence=3](https://sedici.unlp.edu.ar/bitstream/handle/10915/29797/Clase_2013.pdf?sequence=3)
- Fontela, C., & Paez, N. (2015). Programación Orientada a Objetos. Recuperado de [https://www.academia.edu/11537928/Libro\\_programacion\\_orientada\\_a\\_objetos\\_POO\\_](https://www.academia.edu/11537928/Libro_programacion_orientada_a_objetos_POO_)
- Keene, S. E. (1989). Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS. Addison-Wesley. <https://doi.org/10.5555/534810>
- Meyer, B. (1997). Object-Oriented Software Construction (2ª ed.). Prentice Hall. <https://doi.org/10.5555/261119>

## 9. Apéndice

A continuación, adjunto link GitHub de mi repositorio donde se encuentra la actividad Con los archivos completos: [https://github.com/KiwiOwnDaniel27/POO\\_ESPE](https://github.com/KiwiOwnDaniel27/POO_ESPE)