Supuesto 2:
Sistema de Control Automático de la Velocidad de un Vehículo (SCASV)

Autores:

Miguel Medina Ballesteros

Gregorio Vidoy Fajardo

Release Notes.

Para ejecutar el applet:

1. Abrir el proyecto con Eclipse
2. Desplegar la carpeta src
3. Desplegar el paquete interfazGrafica
4. Click derecho en SimuladorControlVehiculo.java > Run as… > Java Applet

Para ejecutar las pruebas Unit:

1. Desplegar la carpeta test
2. Desplegar el paquete que se quiere probar
3. Botón derecho sobre el test > Run as… > JUnit Test
4. En los paquetes con varias clases.
    a. subsistemaDeMonitorizacion
    b. subsistemaDecontrol
5. Botón derecho sobre “TestSuite[NombrePaquete].java” > Run as… JUnit Test

Los niveles monitorizados (Combustible, Aceite, Pastillas de Frenos y Revisión General) están divididos por un factor para que se pueda apreciar mejor el funcionamiento de los niveles (ya que en valores reales jamás se apreciaría que decrece el nivel).

La velocidad del coche es calculada mediante una fórmula en la clase “SimuladorVelocidad”, atendiendo a las leyes físicas de la dinámica.

Documentación.

Diagrama de Casos de Uso

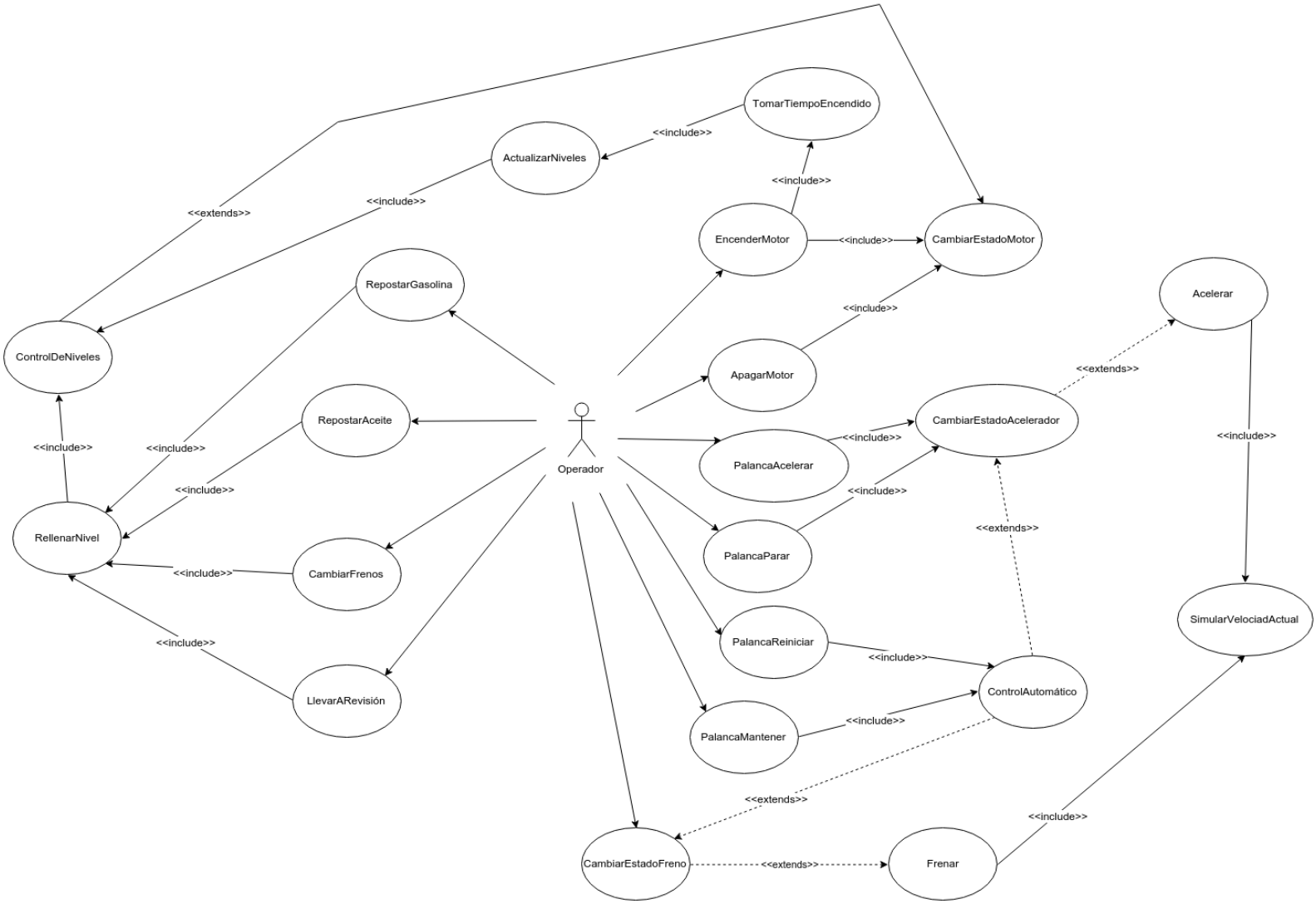

Figura 1. Diagrama de casos de uso

Cada caso de uso directamente utilizable por el Operador se ha acabado correspondiendo a un botón de nuestra interfaz. Al realizarse cada caso de uso el sistema realizará la lógica necesaria para que la acción de control tenga efecto.

Arquitectura Software Utilizada

El diseño arquitectónico que hemos seguido se corresponde con una Arquitectura Orientada a Objetos. Por tanto, hemos perseguido modularizar el comportamiento del coche dividiéndolo en diferentes partes, y el comportamiento del sistema completo dividiéndolo en dos subsistemas:

Subsistema de Monitorización

Es el subsistema encargado de monitorizar el estado del coche y sus componentes en todo momento y comunicarlo a la interfaz para que refresque la pantalla con los datos actualizados periódicamente.

Subsistema de Control

Este subsistema agrupa toda la lógica del funcionamiento del coche. La relación entre el coche y sus componentes más pequeños.

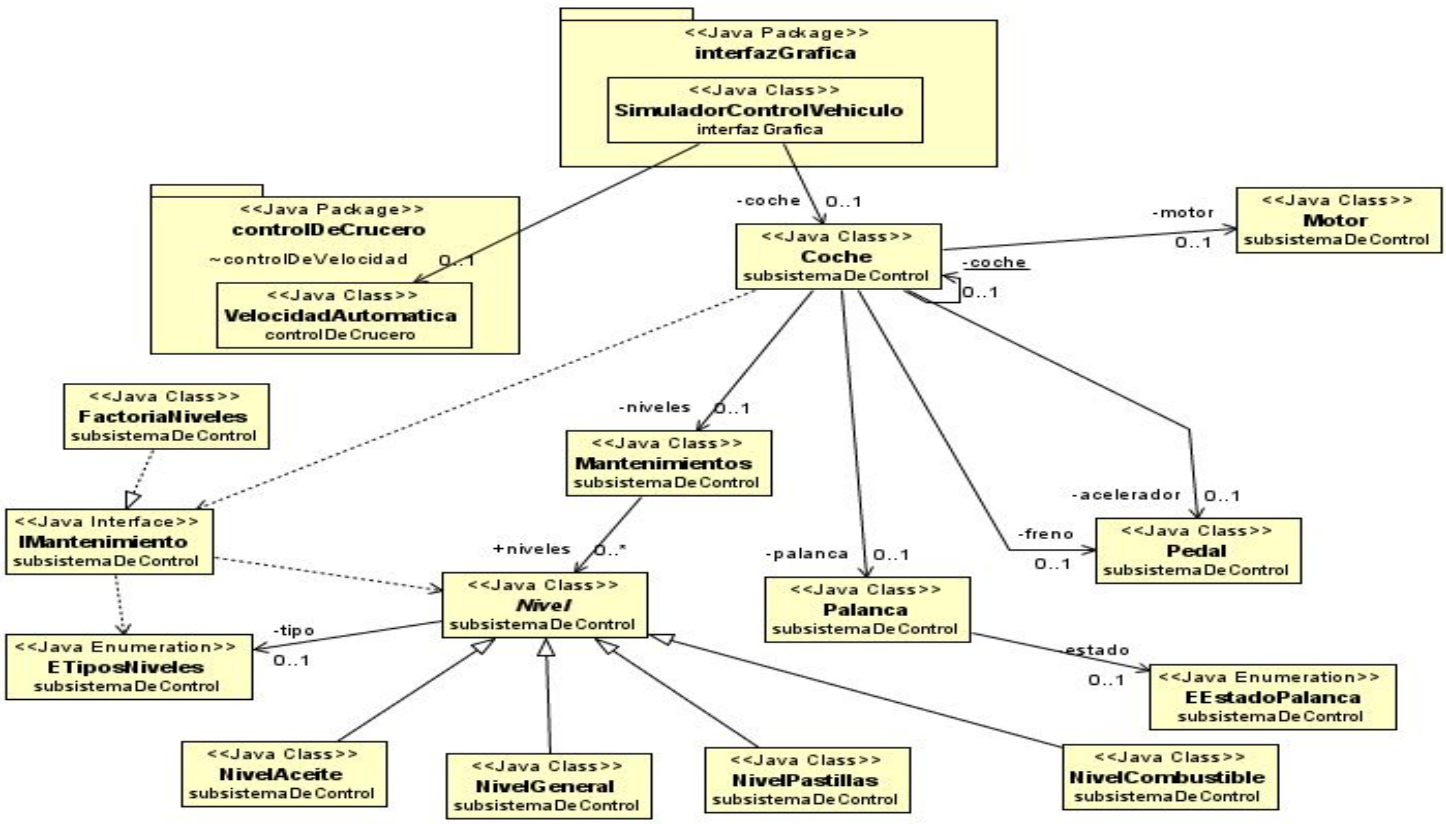

Figura 2. Diagrama UML del Subsistema de Control

Patrones de diseño seguidos

Patrón observable-observador

Para la actualización de los datos de la interfaz hemos seguido el patrón Observable-Observador. El coche es un objeto observable. Como los datos del coche están continuamente cambiando (estado de las palancas, los pedales, el motor, la velocidad, la aceleración, la velocidad crucero, etc), la interfaz debe responder a sus cambios. Para ello, el observable coche notifica a todos sus observadores cuando se realizan cambios en sus valores, solo está encendido. Tenemos un observador por cada valor que tenemos que actualizar. **En definitiva, la comunicación interfaz - coche se realiza mediante este patrón.**

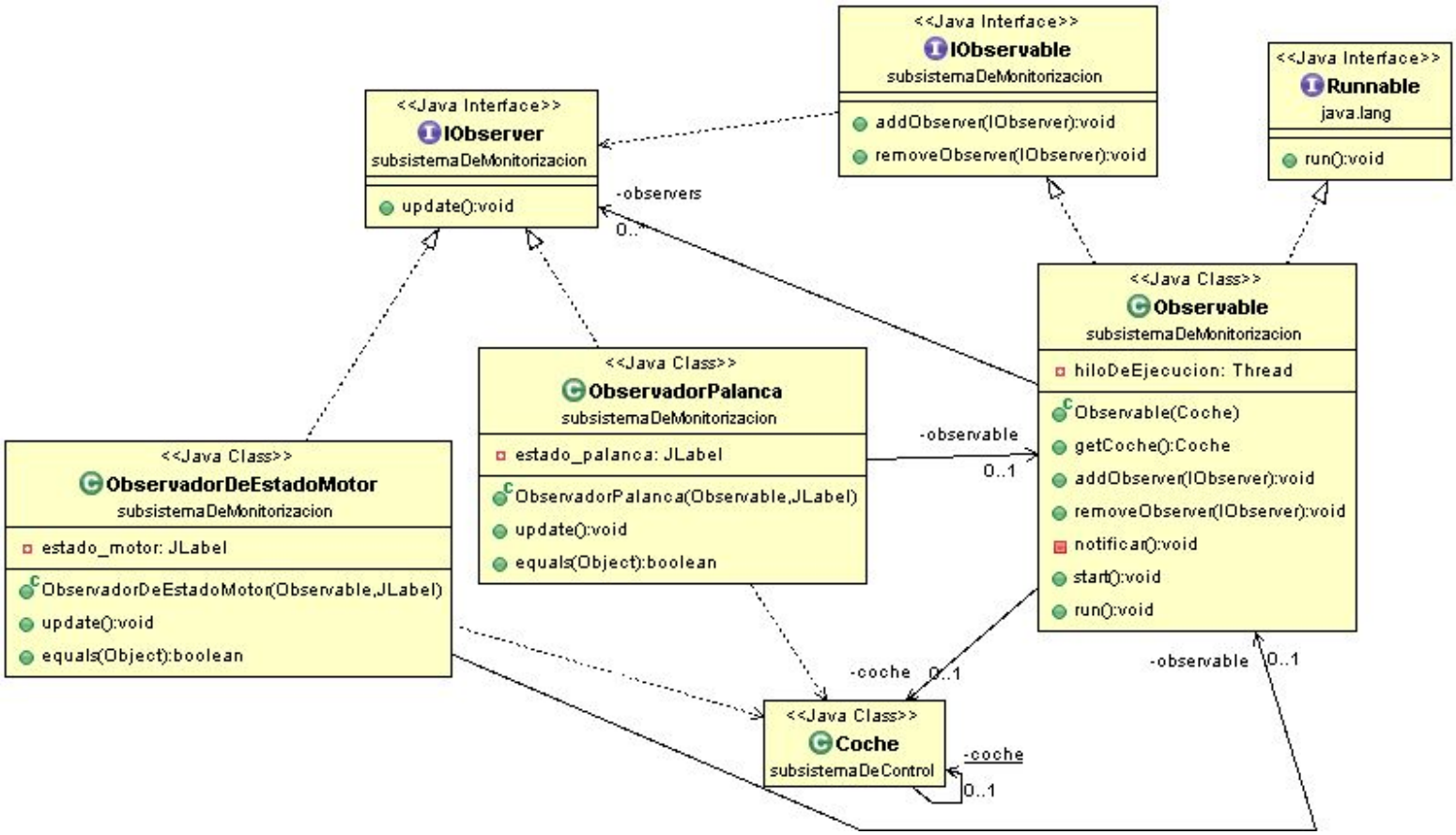

Figura 3. Diagrama UML del patrón Observable-Observador

Patrón factoría

Como son muchos los niveles de mantenimiento que se pueden medir de un coche, y todos ellos se comportan igual, hemos decidido aplicar un patrón factoría para la que la creación de estos se realice de forma dinámica, y así poder abstraernos de su creación y funcionamiento. Abstraemos la creación de cada nivel e instanciamos cada nivel en función del que se pida pasado por parámetro a través de un enum.

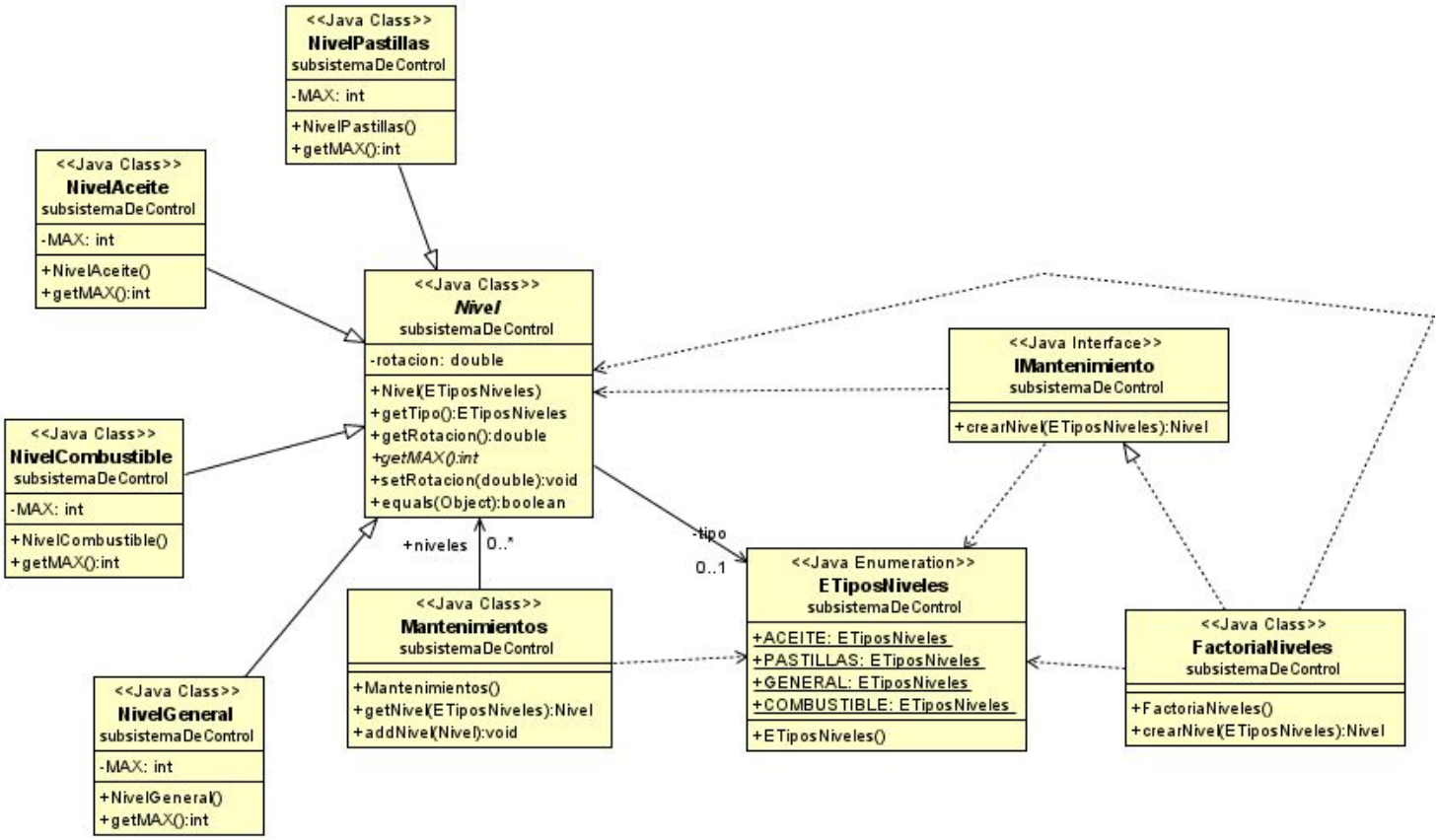

Figura 4. Diagrama UML del patrón Factoría

Diagrama UML Del Sistema Completo

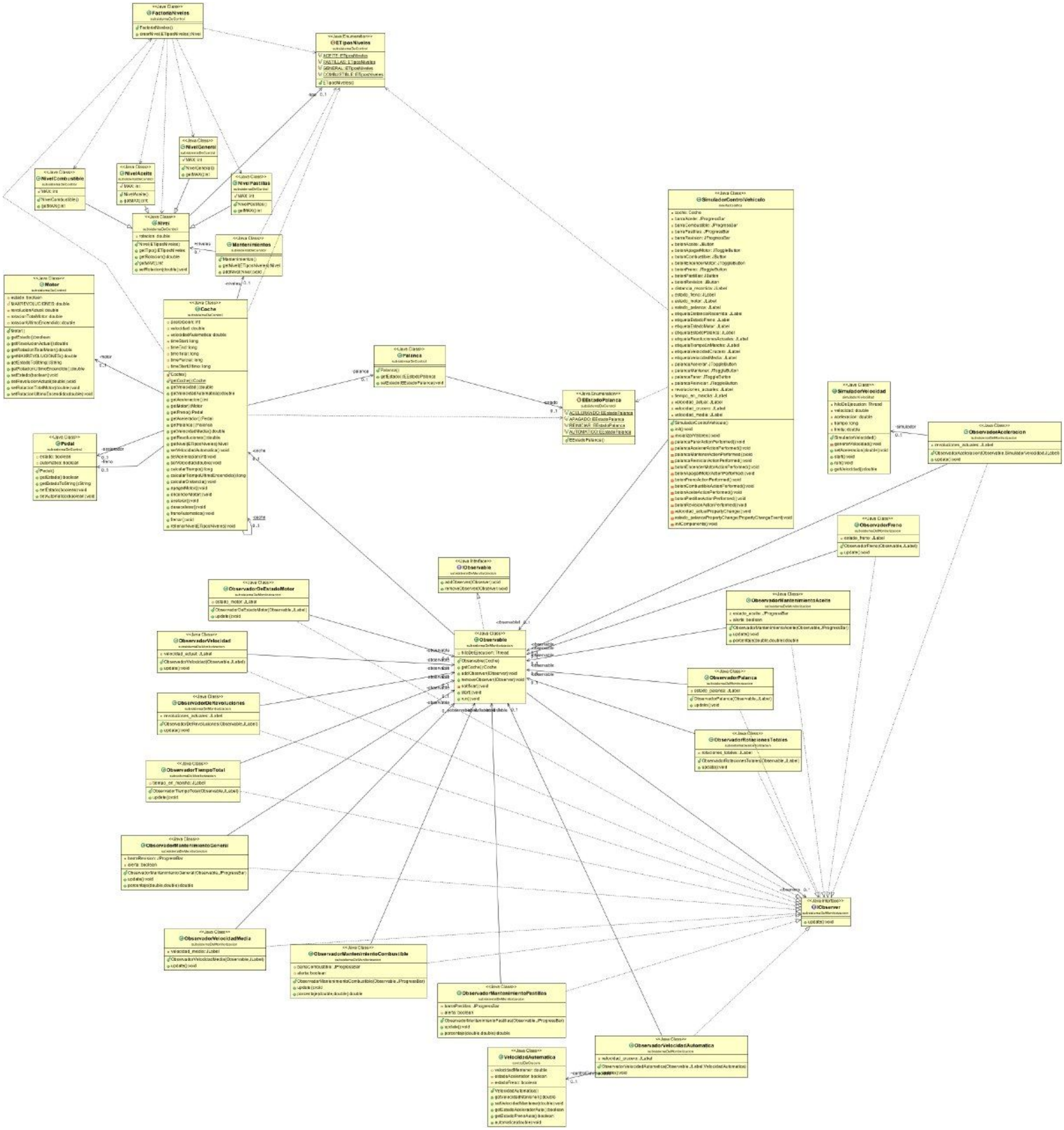

Figura 5. Diagrama UML del Sistema Completo