Práctica 2: SCACV
Simulador de control automático para la conducción de un vehículo

INTRODUCCIÓN

Se realiza el presente documento para describir y justificar las principales decisiones tomadas en cuanto a diseño previo al desarrollo de la práctica 2, en la que se implementa un simulador de control automático de un vehículo de acuerdo con los requisitos funcionales y no funcionales descritos en el guión de la práctica.

Cabe destacar que el mismo se desarrolla en Java y se utiliza la interfaz gráfica swing.

El presente documento constara de los siguientes apartados:

I. ESTILOS ARQUITECTÓNICOS

II. PATRONES DE DISEÑO

III. DIAGRAMA DE PAQUETES

IV. DIAGRAMA DE CLASES

I. ESTILOS ARQUITECTÓNICOS

Se ha seguido el estilo de diseño de **Orientado a Objetos**, ya que de esa forma aumentamos el encapsulamiento, y por tanto aumentamos en independencia y bajo acoplamiento.

Nuestros datos se representan en forma de objetos, y las operaciones que se pueden hacer con estos datos son los métodos de las clases de dichos objetos.

Para el funcionamiento del programa unos objetos tienen referencias a otros objetos y se pasan mensajes entre ellos para ejecutar las distintas funcionalidades del programa.

II. PATRONES DE DISEÑO

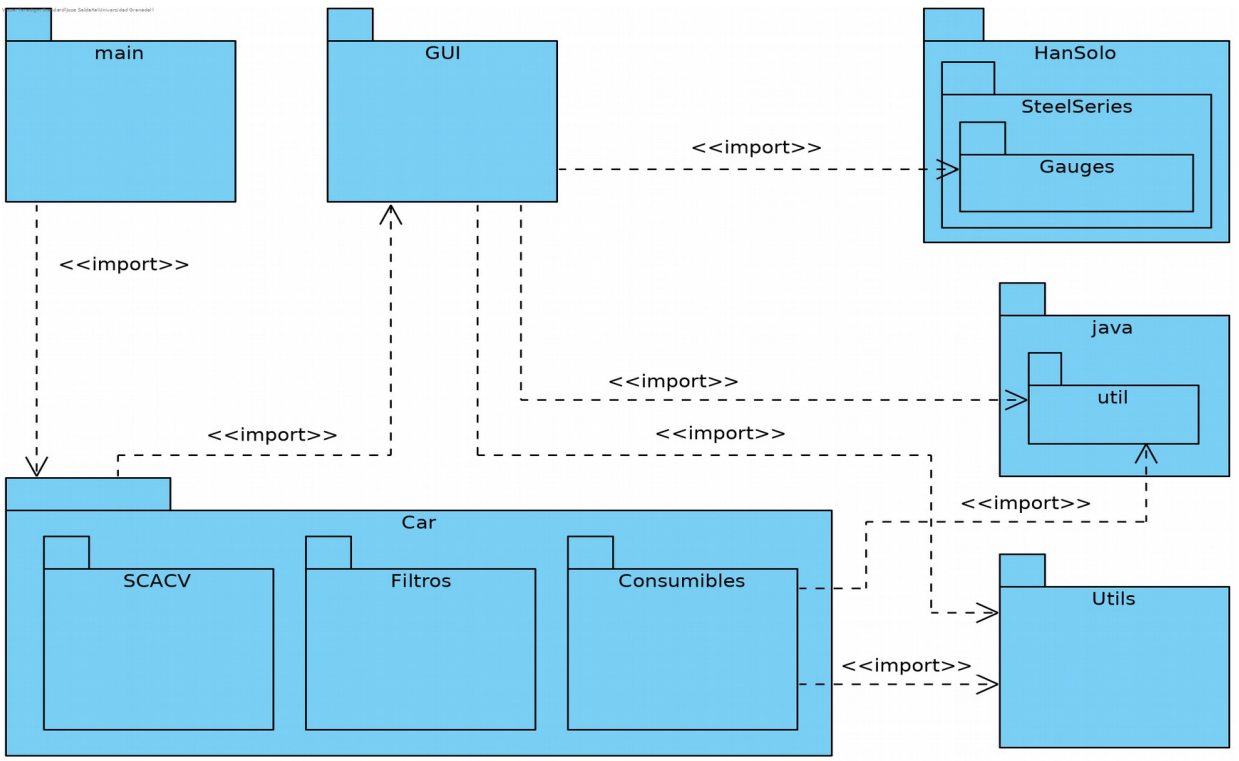
Hemos utilizado el **Patrón Observador** y el **Patrón Filtros de Intercepción**.

Ambos son patrones conductuales, y hemos visto que mediante su uso mejora la calidad del software.

El patrón observador lo usamos para suscribirnos al estado de los consumibles del vehículo, combustible, frenos, aceite o revisión general, y olvidarnos de gestionar manualmente la comprobación de cuando se gasta o es necesario alguno de ellos.
Esto nos permite garantizar la consistencia entre el coche y sus consumibles sin que sea a costa de aumentar el acoplamiento.

El otro patrón utilizado es los filtros de intercepción, y es que el cálculo de la velocidad se lleva a cabo a través de varias operaciones, primero comprobar estado SCACV, posteriormente calcular la velocidad en función del estado del motor y por último repercutir el rozamiento, dichas operaciones son independientes entre sí. El uso de este patrón aumenta la independencia de nuestros componentes, y cada parte del programa se dedica a lo suyo.

III. DIAGRAMA DE PAQUETES



IV. DIAGRAMA DE CLASES