Baltasar Ruiz Hernández 77447873L

Desarrollo de Software

Práctica 2: Supuesto 2:
Desarrollo de un Sistema Completo:
Sistema de Conducción Automático para la Conducción de Vehículos

Explicación del supuesto:

Consiste en desarrollar un sistema de control automático de la velocidad para automóviles. Para ello, cuenta con dos subsistemas principales, que realizan las labores de control y de monitorización, respectivamente:

Subsistema de Control: Realiza las funciones principales de control de los distintos dispositivos que componen el vehículo. Contiene una clase 'singleton' central, llamada *Controlador*, que se encarga de instanciar el resto de componentes del subsistema:

- *DispositivoAcelerador*: Abstracción del acelerador del vehículo. Se caracteriza únicamente por un estado (0 para indicar que el pedal de aceleración no está pisado y 1 para indicar que está activo) y contiene los métodos respectivos para pisar y soltar dicho pedal.

- *DispositivoFreno*: Abstracción del freno del vehículo. Al igual que la clase anterior, tiene un único estado que indica si el pedal está siendo pisado (1) o no (0), junto a los métodos que realizan tal tarea.

- *DispositivoMotor*: Abstracción del motor del vehículo. Funciona de manera similar a las dos clases anteriores, caracterizándose por un estado que indica si el motor está apagado (0) o encendido (1) junto a los métodos necesarios para cambiar entre dichos estados.

- *DispositivoEje*: Abstracción del eje del vehículo. Se caracteriza fundamentalmente por el radio y por métodos de incremento y decremento que se encargan de aumentar o disminuir en mayor o menor medida (dependiendo del argumento de dichos métodos) las vueltas que da el eje.

- *Deposito*: Abstracción del depósito de gasolina del vehículo. Se caracteriza por un valor tope que indica la cantidad máxima que puede almacenar (con el cual se inicializa) y un valor entero que indica el nivel actual del depósito, junto a los métodos necesarios para vaciarlo en cierta cantidad o rellenarlo por completo.

- *DispositivoPalanca*: Abstracción de la palanca que permite viajar entre las distintas funcionalidades que ofrece el sistema de control. Está caracterizado principalmente por el estado actual en el que se encuentra (del tipo enum 'PalancaEstado') así como el método 'moverPalanca' para llevar la palanca al estado deseado (internamente se encarga de comprobar que el estado al que se quiere llegar es correcto teniendo en cuenta el estado actual y de mandar la señal del cambio de estado a la clase controladora).

- *CalculadorVelocidad*: Esta clase se encarga de calcular en todo momento la velocidad actual del vehículo teniendo en cuenta el número de vueltas que ha dado el eje en el intervalo de tiempo que actúa la clase Reloj, que se comenta más adelante.

- *DispositivoAutomatico*: Representa el dispositivo que se encarga de guardar la velocidad a mantener por el vehículo cuando se lleva la palanca al estado 'AUTOMATICO'. En ese momento, el dispositivo se activa (cambia su estado a 1) y en el campo 'velocidadautomatica' guarda la velocidad que en ese momento llevara el vehículo. También dispone del método encargado de olvidar la velocidad a mantener para cuando el usuario apaga el motor.

- *Almacenador*: Clase observable que guarda en todo momento el estado del subsistema de control: estado de la palanca, velocidad actual, velocidad a mantener en el modo automático, nivel del depósito, revoluciones del eje, estado del acelerador y el freno, etc. Sirve además de enlace entre el subsistema de control y el subsistema de monitorización.

- *Controlador*: Tal y como se comentó antes, es la clase central del subsistema, encargada de instanciar todas las clases resumidas previamente y de inicializar los valores de la clase almacenadora. Esta clase está unida a otra clase *Reloj*, la cual periódicamente (cada x milisegundos según indique el campo 'INTERVALO' de dicha clase) llama al método 'accionesPeriodicas' de la clase controladora, en el cual se analiza el estado de los distintos dispositivos y en función de ello actuará de una manera determinada sobre el dispositivo eje, aumentando, disminuyendo o manteniendo la velocidad del vehículo (y en función de ello disminuirá más o menos el nivel del depósito de gasolina). Si en uno de esos instantes de procesamiento el depósito se vaciara por completo o el vehículo alcanzara una velocidad excesiva (más de 250 km/h), se lanzaría una excepción por acceso a un estado ilegal dentro del sistema. La clase controladora dispone además de un método 'recibirSeñal' para cambiar el estado de los dispositivos que corresponda en función del estado al que va a cambiar la palanca ; 'reiniciarVelocidad', para actuar sobre el acelerador o el freno en función de la velocidad actual y la velocidad a la cual se quiere llegar para posteriormente mantener ; 'rellenarDeposito', para actualizar el nivel del depósito al tope inicial y notificar a la clase almacenadora de dicho cambio ; y 'apagarSistema', para llevar todos los valores de los dispositivos y la clase almacenadora al estado inicial, cuando la clase controladora los instanció.

Subsistema de Monitorización: A través de la clase almacenadora, se encarga de calcular algunos valores para así monitorizar en todo momento el SCACV. Al igual que el subsistema de control, dispone de una clase 'singleton' central llamada *ControladorMonitorizador* que instancia los componentes necesarios para llevar a cabo la funcionalidad presentada:

- *MonitorizadorCombustible*: Teniendo en cuenta el valor del depósito actual y una variable en la que guarda el valor que tenía previamente, puede calcular la diferencia

y añadirla así a una variable 'gastototal'. De esta manera, la clase sabe en todo momento cuánto combustible ha gastado en total a lo largo del tiempo de monitorización, pudiendo calcular así el gasto medio. Dispone además de un método de reinicio para llevar de nuevo el contador total de gasto a 0, para cuando el usuario apaga el motor.

- *MonitorizadorMotor*: Se encarga de calcular la velocidad media del vehículo teniendo en cuenta el número total de revoluciones del eje a lo largo del período de monitorización. Dispone además de un método de reinicio para llevar de nuevo el contador total de revoluciones a 0, para cuando el usuario apaga el motor.

- *Mantenimiento*: Es la clase encargada de monitorizar el cambio de aceite, pastillas y la revisión general del sistema. Para ello, de manera similar a la clase *MonitorizadorCombustible*, teniendo en cuenta el valor actual de revoluciones y el valor anterior calcula la diferencia entre ambos, con la cual actualiza un contador que al llegar a cierta cantidad significa la necesidad de realizar una de las tres tareas de mantenimiento (p. ej. al llegar a 5 millones de revoluciones indica un cambio de aceite necesario). Al llegar a este momento, la clase utiliza a la clase auxiliar *NotificadorMantenimiento* para guardar en la clase almacenadora *AlmacenadorMonitorizador* (comentada más adelante) la necesidad de realizar tal tarea. Dispone además de un método 'arreglarVehiculo' con el cual (utilizando de nuevo a la clase auxiliar) guarda en la clase almacenadora el fin de la tarea de mantenimiento así como el número de revoluciones del sistema en el momento en el que dicha tarea terminó.

- *AlmacenadorMonitorizador*: De manera similar a la clase almacenadora del subsistema de control, guarda en todo momento los cálculos y estados procedentes del subsistema de monitorización: tres variables enteras que indican la necesidad (1) o no (0) de realizar una de las tres tareas de mantenimiento existentes y otras tres variables para guardar las revoluciones del sistema cuando la tarea de mantenimiento correspondiente fue realizada.

- *ControladorMonitorizador*: Es la clase central del subsistema, encargada de instanciar los componentes mencionados previamente. Además, al igual que su homóloga en el subsistema de control, esta clase va asociada a una clase *Reloj* para llevar a cabo las labores contenidas en el método 'accionesPeriodicas' cada cierto tiempo: calcular la velocidad y el gasto de combustible promedio (los cuales he creído mejor guardar en la clase almacén del subsistema de control en vez de la clase *AlmacenadorMonitorizador*) y actualizar los valores del objeto de la clase *Mantenimiento* para proceder llegado el caso (ya explicado en su sección).

GUI: Contiene las clases encargadas de crear la interfaz gráfica del SCACV para el usuario. Aquí es donde se declara la clase padre *Observable* (extendida por las dos clases almacenadoras) así como la interfaz *Observador* y todas las clases observadoras que la implementan. Cada una de estas clases observadoras posee un

atributo de enlace a uno u otro almacén así como una serie de atributos en función del propósito para el cual fueron creadas (p. ej. la clase observadora del depósito se encarga de 'observar' el nivel actual del depósito y el gasto medio). Cada vez que una clase almacenadora ha cambiado su estado a raíz de la llamada a cualquiera de sus métodos 'almacenarX', notifica a las clases que la estén observando dicho cambio para que estas también actualicen su estado.

Además, la GUI dispone de 4 paneles secundarios y un panel principal:

- *PanelVelocidad*: Muestra al usuario la velocidad actual, la velocidad media y la velocidad a mantener por el vehículo en caso de llevar la palanca al estado 'REINICIANDO'. Los valores se mostrarán en km/h.

- *PanelDepósito*: Muestra al usuario el porcentaje de combustible restante, así como una barra de progreso que va vaciándose y cambiando progresivamente del azul al rojo en función del nivel del depósito, y el gasto medio de combustible en litros por hora.

- *PanelInformativo*: Muestra al usuario el estado actual de la palanca y el estado (ON u OFF) del acelerador y el freno. Además, dispone de una sección en la cual hay tres botones (no disponibles inicialmente) que se activan cuando el vehículo requiere la tarea de mantenimiento asociada a dicho botón, momento en el cual el botón vuelve a quedar inactivo hasta que la tarea es requerida de nuevo y se muestra un mensaje indicando las revoluciones realizadas por el coche en el momento que se pulsó el botón.

- *PanelBotones*: Contiene los 7 botones con los cuales el usuario puede cambiar entre los diferentes estados de la palanca, rodeando con un recuadro rojo el botón asociado al estado actual de la palanca en el sistema. Todos ellos son de una única pulsación excepto el botón de frenar, el cual actuará únicamente mientras el usuario mantenga el ratón pulsado sobre él. Hay además un botón extra con el cual se puede rellenar el depósito y que funciona exclusivamente cuando el motor está apagado.

- *PanelSimulador*: Es el panel principal del sistema, el que contiene las instancias de los dos controladores y el que se encarga de instanciar los diferentes observadores y paneles que ya se resumieron anteriormente. Está también asociado a una clase *Reloj* con la cual llama cada cierto tiempo a los métodos periódicos que actualizan todos y cada uno de los componentes (barra de progreso, etiquetas, botones, etc.) de los paneles secundarios.

NOTAS:

- Las imágenes con los diagramas de clases podrán encontrarse en el mismo directorio que este 'pdf', donde serán mejor visualizadas que insertándolas en este archivo.

- Los sonidos utilizados en la aplicación han sido tomados como 'mp3' desde la página libre http://www.freesfx.co.uk/ y pasados posteriormente a 'wav'.

- La programación de la aplicación ha sido realizada con el IDE Eclipse for Java Developers Version Mars.2 Release (4.5.2) y los diagramas han sido realizados con Visual Paradigm Community Edition 13.0.