Módulo 1: Java

Jorge Pastor García

NIA: 100315924

1. Objetivo

En esta práctica se pide realizar un programa de gestión de un parking cuyos requisitos principales son:

- Dar de alta un vehículo.
- Buscar un vehículo.
- Retirar un vehículo.
- Cambiar el coste por minuto.
- Guardar y modificar ficheros de texto.

Además, se pide que el programa cumpla otras especificaciones como que el programa no sea sensible a mayúsculas ni minúsculas, o el uso de interrupciones entre otras.

2. Planteamiento

El programa implementado consta de tres clases:

- Coche:
  Donde defino los atributos del vehículo. (Matricula, Modelo, Color, Titular, Fecha y Horas de entrada). Así como sus métodos que serán gets y sets para inicializar los atributos del coche desde la aplicación.

- Menú:
  En la clase menú implemento los menús necesarios para mi programa. Aunque es innecesario he decidido incluirlo para aumentar el orden y la comprensión.

- Aplicación:
  Es la clase donde se desarrollo los métodos necesarios para implementar el programa solicitado. Esta clase permite dar de alta y mostrar usuarios, así como cambiar el coste por minuto y guardar los datos en los ficheros.

En el main tan solo se llama a la función que muestra el menú principal, desde esa función se llaman a distintos métodos de la clase aplicación según se requiera.

Los datos se guardan en un vector dinámico de tipo de dato coches, en el que cada elemento del vector será in vehículo dado de alta en el sistema. Cuando un coche se da de baja se borra el elemento en el que estaba guardado del vector. Posteriormente, los datos del vector se irán guardando en el fichero de texto cada vez que se haga una modificación. Y cada vez que se entre en la aplicación se cargaran los datos desde el fichero de texto.

Además, he decidido para hacer más cómoda la interacción con el programa, solicitar la entrada de datos a través de cuadros de diálogos a través de la opción JOtionPane que nos ofrece Java.

Los datos de salida del sistema se mostrarán por consola. Creo que esta opción hace más visual y llevadero el programa.

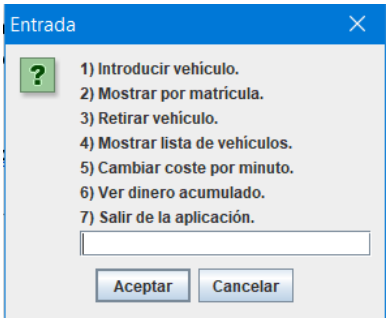

Figura1. Cuadro de diálogos.

3. Clase Aplicación

En la clase aplicación se implementan los métodos más importantes, por lo que procederé a explicarlos:

- **Introducir vehículo:**
  En esta función se piden los datos necesarios para inicializar los atributos de la clase coche, forzando a introducir aquellos campos que sean obligatorios como son Matricula de entrada y Fecha de entrada. La fecha y hora de entrada se registran automáticamente por el sistema y se convierten a string para almacenarlo en el vector de coches.

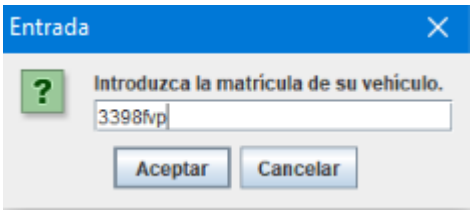

Figura2. Solicitud de matricula.

- **Mostrar vehículo por matricula:**
  En esta función se solicita una matrícula a introducir por el usuario, si la encuentra mostrará los datos del vehículo solicitado, y por último nos dará la opción de retirar el vehículo. Si elegimos retirarlo se llamará a la función que calcula el dinero a pagar según el tiempo de estacionamiento.

```
***LOS DATOS DE LA BUSQUEDA SON***

Matricula: 0001ATB
Modelo: BMW SERIE1
Color: BLANCO
Titular: ANA CAMPOS
Fecha de entrada: 19:11:2017
Hora de entrada: 21.08
```

Figura3. Datos solicitados por matricula.

- **Retirar vehículo:**
  En esta función se muestran todos los vehículos registrados en el sistema. Los vehículos están ordenados por orden de entrada al sistema. Cuando el usuario solicita esta opción debe de elegir un numero de la lista para retirar el vehículo. Si se desea volver al menú principal tan solo hay que dar a aceptar o a cancelar y no se realizará ninguna acción.
  En caso contrario se elimina el vehículo seleccionado del sistema.

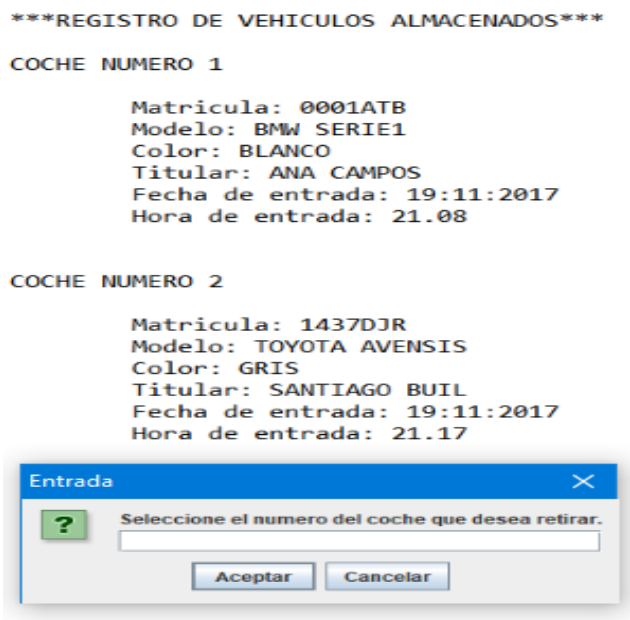

Figura4. Retirar vehículos por número.

- **Mostrar vehículos:**
  En esta opción se muestran todos los coches registrados en el sistema. Para ello se recorre el vector dinámico y se recuperan los elementos de cada coche uno a uno imprimiéndolos.

- **Cambiar coste por minuto:**
  Esta variable se inicializa desde un fichero, a fin de que no tengamos que cambiarla del código cada vez que entremos en la aplicación. Cada vez que se cambie, la variable donde se guarda el coste por minuto se sobrescribe en el fichero y permanecerá invariante hasta que se vuelva a cambiar a pesar de que salgamos del programa.

- **Calcular coste por minuto:**
  Esta función recupera la fecha y la hora de entrada del vehículo seleccionado en forma de string, tras recuperar ambos parámetros se juntan en un nuevo string con el formato que hayamos elegido del tipo Date.
  Cuando retiramos un vehículo se llama a esta función, que registra la hora de salida del vehículo seleccionado.
  Cuando tenemos tanto la fecha de entrada, como la fecha de salida tan solo tenemos que restar ambos tipos de datos. El resultado lo obtenemos en milisegundos y con una conversión a minutos ya tenemos el tiempo de estacionamiento.

Tan solo queda multiplicarlo por el coste por minuto para obtener el coste de estacionamiento.

```
public void Calcular_Precio(int cont) throws java.text.ParseException{

    Date Fecha_Entrada = null;  //Creo un tipo date para recuperar la fecha de entrada
    Date Fecha_Salida = null;   //Cuando entro en esta función creo un tipo date registrando la hora de salida

    Calendar date = Calendar.getInstance();

    String dia_entrada1 = Listado_Coche.elementAt(cont).getFecha(); //Recupero la fecha de entrada en forma de string
    String hora_entrada1 = Listado_Coche.elementAt(cont).getHora(); //Recupero la hora de entrada en formato string
    String entrada = dia_entrada1 +" "+hora_entrada1;   //Junto fecha y hora en un solo string


    SimpleDateFormat fecha_entrada = new SimpleDateFormat("dd:MM:yyyy HH.mm");  //Defino el formato de la fecha de entrada
    Fecha_Entrada = fecha_entrada.parse(entrada);   //Convierto el string fecha de entrada a tipo date
    Fecha_Salida = date.getTime();                  //Registro la hora de salida

    long tiempo_ms = Fecha_Salida.getTime() - Fecha_Entrada.getTime();  //La fecha de salida menos la de entrada es el tiemo que ha permanecido en el parking
    long minutos = tiempo_ms/(1000*60);         //Paso los milisegundos que ha estado en elparking a minutos

    dinero_pagar = coste_minuto * minutos;  //Calculo el dinero que ha costado el estacionamiento

    JOptionPane.showMessageDialog(null , "Usted ha estado estacionado "+minutos+" minutos.\n\nEl dinero que debe desembolsar es de "+dinero_pagar+" Euros.");

}
```

Figura5. Cálculo del coste de estacionamiento.

- **Actualizar Fichero:**
  Cada vez que se introduce un vehículo o se retira, se actualizan los datos del parking. Para ello se llama a esta función que actualiza todos los datos del parking.

- **Inicializar Parking:**
  Esta función recorre el fichero línea a línea de forma independiente. En cada línea busca los diferentes puntos y comas almacenado la posición de inicio y fin de cada campo y almacenándolas.
  Después mediante el método substring se recuperan campo a campo los distinto elementos y se vuelven a guardar en la clase vector. Esta función se llama al principio del programa para recuperar los vehículos nada más acceder a la aplicación.

- **Calcular dinero acumulado:**
  Cada vez que se devuelve un coche, se realiza el cálculo del dinero acumulado en el sistema. Para ello se carga desde un fichero una variable con el total. A esta variable se le sumara el precio que ha pagado cada vehículo devuelto.

- **Guardar dinero acumulado:**
  Cada vez que se retira un vehículo se llama a la función calcular dinero, que a su vez llamará a esta función que escribe el dinero acumulado en un fichero para poder llevar un cálculo del total recaudado.

- **Inicializar dinero acumulado**
  Cada vez que se acceda al sistema se cargaran estos datos al inicio del programa. Para poder llevar el registro a pesar de salir de la aplicación.

- **Mostrar dinero acumulado:**
  Esta función mostrara el dinero recaudado por el sistema gracias al fichero de texto previamente cargado. Adicionalmente se da la opción de resetear el registro poniéndolo a cero a gusto del administrador.

- **Coste por minuto:**
  El coste por minuto se carga desde un fichero de texto, así se evitará cambiarlo cada vez que se acceda al programa si es un valor diferente al asignado por defecto. Con este valor se calcula el coste de estacionamiento de cada vehículo devuelto.

4. Clase menú, coche y main

En la clase coche tan solo se implementan los métodos para inicializar los atributos del objeto coche mediante los gets y sets (getMatricula(), setMatricula(), setModelo(), getModelo()…). A estos métodos se les llamará desde la clase aplicación para poder manipular los atributos privados.

En la clase Menú se llama a los métodos necesarios para el funcionamiento de la aplicación. Esta clase no es necesaria, pero sí ayuda a una mejor estructura y orden en el programa.

En el main solo se crea el objeto menú y se llama al menú principal.

5. Formato del fichero de coches registrados