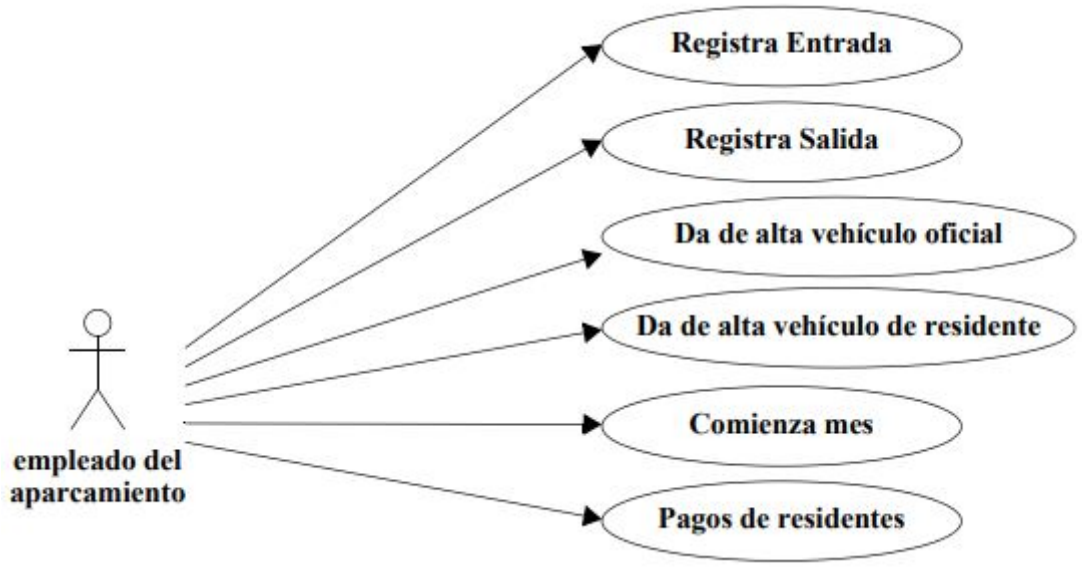
Ejercicio completo:
Se desea gestionar el acceso de vehículos a un aparcamiento de pago. El aparcamiento no se encuentra automatizado, por lo que existe un empleado encargado de registrar las entradas y salidas de vehículos. Los vehículos se identifican por su matrícula. Cuando un vehículo entra en el aparcamiento el empleado registra su entrada y al salir registra su salida y, en algunos casos, cobra el importe correspondiente por el tiempo de estacionamiento.

El importe cobrado depende del tipo de vehículo:
• Los vehículos oficiales no pagan, pero se registran sus estancias para llevar el control. (Una estancia consiste en una hora de entrada y una de salida)
• Los residentes pagan a final de mes a razón de 0.002€ el minuto. La aplicación irá acumulando el tiempo (en minutos) que han permanecido estacionados.
• Los no residentes pagan a la salida del aparcamiento a razón de 0.02€ el minuto. Se prevé que en el futuro puedan incluirse nuevos tipos de vehículos, por lo que la aplicación desarrollada deberá ser fácilmente extensible en ese aspecto.

Casos de uso:



A continuación se procede a describir los casos de uso. No se entra en detalles de la interacción entre el empleado y la aplicación (punto 1 de cada caso de uso), puesto que no va a ser tarea del alumno desarrollar esa parte.
Caso de uso "Registra entrada":
1. El empleado elige la opción "registrar entrada" e introduce la matrícula del coche que entra.
2. La aplicación apunta la hora de entrada del vehículo.
Caso de uso "Registra salida":
1. El empleado elige la opción "registrar salida" e introduce la matrícula del coche que sale.
2. La aplicación realiza las acciones correspondientes al tipo de vehículo: - oficial: asocia la estancia (hora de entrada y hora de salida) con el vehículo

- residente: suma la duración de la estancia al tiempo total acumulado
- no residente: obtiene el importe a pagar

Caso de uso "Da de alta vehículo oficial":
1. El empleado elige la opción "dar de alta vehículo oficial" e introduce su matrícula.
2. La aplicación añade el vehículo a la lista de vehículos oficiales

Caso de uso "Da de alta vehículo de residente":
1. El empleado elige la opción "dar de alta vehículo de residente" e introduce su matrícula.
2. La aplicación añade el vehículo a la lista de vehículos de residentes.

Caso de uso "Comienza mes":
1. El empleado elige la opción "comienza mes".
2. La aplicación elimina las estancias registradas en los coches oficiales y pone a cero el tiempo estacionado por los vehículos de residentes.

Caso de uso "Pagos de residentes":
1. El empleado elige la opción "genera informe de pagos de residentes" e introduce el nombre del fichero en el que quiere generar el informe.
2. La aplicación genera un fichero que detalla el tiempo estacionado y el dinero a pagar por cada uno de los vehículos de residentes.

El formato del fichero será el mostrado a continuación:

```
Matrícula     Tiempo estacionado (min.)    Cantidad a pagar
S1234A              20134                        40.27
4567ABC              4896                         9.79
...                  .....                        .....
```

La aplicación contará con un programa principal basado en menú que permitirá al empleado interactuar con la aplicación (dicho programa principal se supone encargado a otro programador, por lo que no deberá ser realizado por el alumno).

El alumno deberá desarrollar las clases que permitan gestionar los vehículos con sus datos asociados (estancias, tiempo, ...), las listas de vehículos registrados como oficiales y residentes, etc.

Se pide:

• diseño arquitectónico de la parte de la aplicación encargada al alumno
• código de las clases correspondientes a la parte de la aplicación encargada al alumno

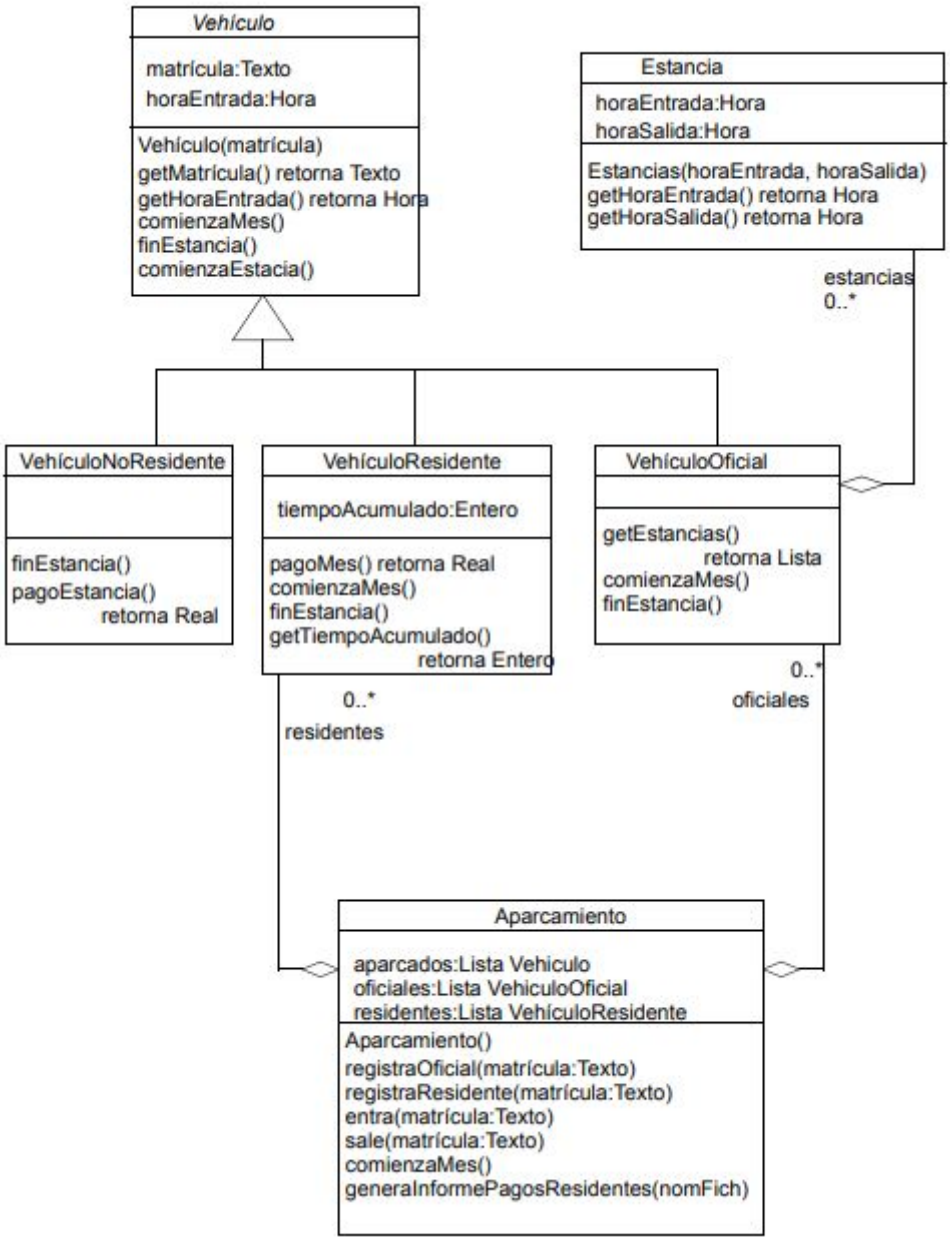
Para obtener la fecha y hora actual se utiliza la clase `Calendar`:

```
Calendar unaFecha; // para almacenar una fecha
...
unaFecha=Calendar.getInstance(); // obtiene la fecha actual
```

Para obtener intervalos de tiempos entre dos fechas suponer que se dispone del método:

```
/** Obtiene la diferencia en minutos entre dos fechas
 * @param inicial fecha inicial
 * @param final fecha final
 * @return diferencia final-inicial en minutos
 */
private static int difEnMinutos(Calendar inicial,
                                Calendar final) {...}
```

Diseño arquitectónico:



Nota:
La clase Vehiculo la definiremos abstracta ya que los metodos comienzaMes() y finEstancia() son abstractos.

VehiculoNoResidente, tiene como atributo:
private static final double precioMinuto
private double pagoEstancia

VehiculoResidente,
private static final double precioMinuto
private int tiempoAcumulado

En la solución propuesta por el autor se utiliza LinkedList en vez de arraylist
Ver el siguiente link:
http://www.enrique7mc.com/2016/07/diferencia-entre-arraylist-y-linkedlist/

Te animas a Codificar el diagrama de clases?