Empresa de Alquiler de Vehículos

version 1.0 - 14/05/2021

Sobre la práctica

Leer con atención los siguientes puntos:

1. Esta práctica es individual.
2. Se pueden compartir ideas, pero no código. Las prácticas copiadas significarán un cero para el copiador y para el copiado, en esta práctica y en todas las prácticas del curso.
3. El alumno sólo deberá entregar los archivos que aparecen en el apartado *Material a entregar*.
4. Entregar solamente archivos con extensión .java. No entregar archivos comprimidos ni archivos .class.
5. Los nombres de los archivos de programa (clases) tienen que ser los que se dan, así como los de las operaciones.
6. No entregar archivos vacíos.
7. Poner el nombre del autor del programa en todos los archivos.
8. Cualquier incumplimiento de los apartados 3 al 8 significará un cero en la práctica.
9. Respetar la estructura de paquetes que se da.

1. Descripción general

1.1. Los vehículos

Se pretende desarrollar una aplicación para una empresa de alquiler de vehículos.
El objetivo de la aplicación es dar de alta a los vehículos que llegan, atender peticiones de clientes y calcular los precios de los alquileres.
Cada vehículo se identifica unívocamente por el número de su matrícula.
La empresa alquila distintos tipos de vehículos, tanto para transporte de personas como de carga. En la actualidad, los vehículos alquilados por la empresa son: coches, microbuses, furgonetas de carga y camiones.
En la empresa se dan de alta vehículos que llegan procedentes de diferentes orígenes y se clasifican.
El precio del alquiler de cualquier vehículo tiene una componente base que depende de los días de alquiler a razón de 50€/día.
En el caso de alquiler de un coche, al precio base se le suma la cantidad de 1.5€ por plaza y día.
El precio de alquiler de los microbuses es igual que el de los coches, salvo que se le añade una cantidad de 2€/plaza independientemente de los días de alquiler.
El precio de los vehículos de carga es el precio base más 20€ * PMA (PMA = peso máximo autorizado en toneladas).
Además, en el caso de los camiones, al precio se suma un fijo de 40€ independientemente de los días de alquiler.
Un cliente viene identificado por su DNI.

1.2. Las peticiones y la empresa

Una petición de un vehículo consta del cliente que la hace, el código del tipo de vehículo que se solicita y el número de días que se quiere alquilar el vehículo.
Se dispone de una colección de vehículos, *vehiculos*, en la cual están todos los vehículos que tiene dados de alta la empresa ordenados por orden de llegada. *vehiculos* será un objeto de una clase llamada *Vehiculos*. Esta clase será implementada mediante una lista.

Se dispone también de cuatro colas, una para cada tipo de vehiculos, a donde se van pasando los vehículos desde *vehiculos* por orden de llegada. Estas cuatro colas están guardadas en una lista llamada *flota*. La aplicación gestiona el alta clasificando cada vehículo en su sitio correspondiente de la flota.
Se dispone de una cola de peticiones llamada *peticiones*, en la cual están todas las peticiones que hacen los clientes. Cada petición viene con el DNI del cliente, un código de tipo de vehículo pedido (0=coche, 1=microbús, 2=furgoneta, 3=camión) y el número de días que se quiere alquilar el vehículo.
También se tiene una colección de vehiculos *asignados* implementada mediante una lista, al igual que *vehiculos*, por lo que ésta puede ser otro objeto de la clase *Vehiculos*.
Por último se tiene una *listaDeEspera* de peticiones que no han podido ser resueltas por no haber ningún vehículo disponible del tipo pedido. Esta lista de espera se implementa mediante una cola.
Así pues, la empresa se compone de:

- vehiculos.
- flota.
- peticiones.
- asignados.
- listaDeEspera.

1.3. Casos de uso

Las operaciones que se deben poder realizar en la empresa son las que aparecen en los siguientes casos de uso.

1.3.1. Caso de uso *Carga de Datos* (constructor de Empresa y cargarFlota).

Se cargan los *vehiculos* y las *peticiones* y se cargan los *vehiculos* en *flota*.
En la práctica esto se hará en el constructor de Empresa, el cual utilizará como método auxiliar *cargarFlota*.

1.3.2. Caso de uso *Alquilar Vehículos* (operación alquilarVehiculos).

La aplicación va procesando la cola de peticiones una a una y va asignando para cada petición el primer vehículo de los disponibles en *flota* del tipo que se haya pedido. Cuando se asigna, se borra de una de las colas de *flota* y se pone en *asignados*. Si no hay disponibles de ese tipo se guarda el vehiculo en la lista de espera.

1.3.3. Caso de uso *Mejor Cliente* (operación mejorCliente).

Se obtiene el cliente (su DNI) que más peticiones de vehículos ha hecho. Este dato se puede obtener de las *peticiones* antes de que hayan sido procesadas.
Las especificaciones más detalladas de los datos se dan a continuación en la sección Modelo de Datos.

2. Modelo de datos.

2.1. Estructura de paquetes.

Todas las clases de las secciones 3 y 4 irán en el paquete vehiculos.
Todos los testers se incluyen en el paquete pruebas.
Las clases Petición y Empresa van en el paquete empresa.

2.2. Estructura y funcionalidad de las clases.

```
Vehiculo
+ static final int COCHE = 0, MICROBUS = 1, FURGONETA = 2, CAMION = 3
+ static final double PRECIO_POR_DIA = 50.0
- String matricula   + Vehiculo (String matricula)
+ double precioAlquiler (int dias)
+ String matricula ()
+ String toString ()

VehiculoPasajeros extends Vehiculo
+ static final double PRECIO_POR_PLAZA_Y_DIA = 1.5
- int plazas
+ VehiculoPasajeros (String matricula, int plazas)
+ double precioAlquiler (int dias)
+ int plazas ()
+ String toString ()

Coche extends VehiculoPasajeros
+ Coche (String matricula, int plazas)
+ String toString ()

Microbus extends VehiculoPasajeros
+ static final double PRECIO_POR_PLAZA = 2
+ Microbus (String matricula, int plazas)
+ double precioAlquiler (int dias)
+ String toString ()

VehiculoCarga extends Vehiculo
+ static final double PRECIO_POR_TONELADA_DE_PMA = 20
- double pma
+ VehiculoCarga (String matricula, double pma)
+ double precioAlquiler (int dias)
+ String toString ()

Furgoneta extends VehiculoCarga
+ Furgoneta (String matricula, double pma)
+ String toString ()

Camion extends VehiculoCarga
+ Camion (String matricula, double pma)
+ double precioAlquiler (int dias)
+ String toString ()

Vehiculos extends ArrayList<Vehiculo>
+ Vehiculos ()
+ void registrarVehiculo (Vehiculo v)
+ Vehiculo buscaVehiculo (String matricula)
+ double precioAlquiler (String matricula, int dias)
+ String mostrarTodosLosPrecios (int dias)

Peticion
- String dni  //DNI del cliente que hace la petición.
```

```
- int codigo  //código del Vehiculo a alquilar (ver códigos en clase Vehiculo).
- int dias
+ Peticion (String dni, int codigo, int dias)
+ String dni ()
+ int codigo ()
+ Peticion clone ()
+ String toString ()

Empresa
- Vehiculos vehiculos
- IList<IQueue<Vehiculo>> flota
- IQueue<Peticion> peticiones
- Vehiculos asignados
- IQueue<Peticion> listaDeEspera
- static Vehiculos cargaVehiculos (Vehiculo[] vs)
- static IQueue<Peticion> cargaPeticiones (Peticion[] ps)
+ Empresa (Vehiculo[] vs, Peticion[] ps)
+ void cargarFlota ()
+ void alquilarVehiculos ()
- int posicion (String dni, IList<Contador> lf)
- void actualizar (String dni, IList<Contador> lf)
+ String mejorCliente ()
+ String toString ()
```

2.3. Relaciones de contenido y de herencia entre las clases del paquete vehiculos.

Estas relaciones están representadas gráficamente en el diagrama UML de la figura 1.

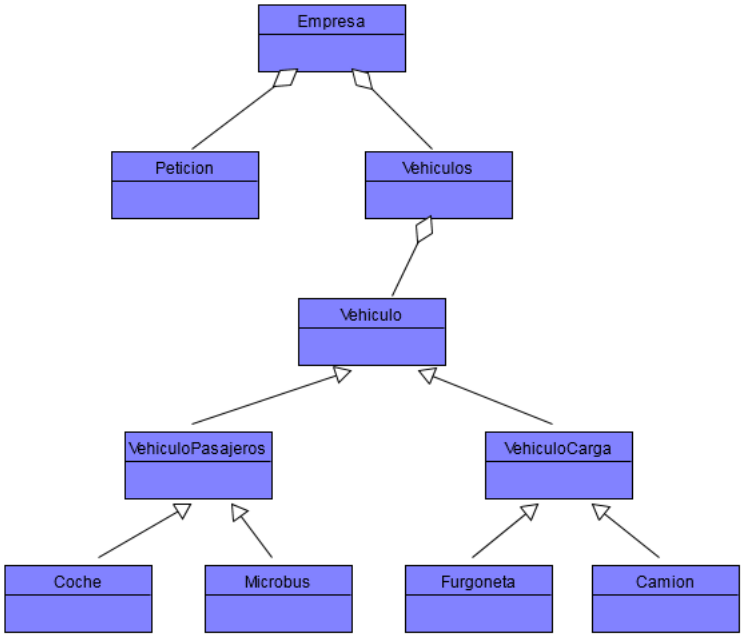

Figura 1: Diagrama UML de clases.

3. Clase *Vehiculo* y sus derivadas: VehiculoPasajeros (Coche y Microbus), y VehiculoCarga (Furgoneta y Camion).

Material necesario

- VehiculoTest.java

Material a entregar

- Vehiculo.java
- VehiculoPasajeros.java
- Coche.java
- Microbus.java
- VehiculoCarga.java
- Furgoneta.java
- Camion.java

Se dispone del tester VehiculoTest para probar todas las clases pedidas en esta sección.
Todas estas clases van dentro del paquete vehiculos.

4. Clase *Vehiculos*.

Material necesario

- VehiculosTest.java
- Vehiculos.java

Material a entregar

- Vehiculos.java

La clase Vehiculos hereda de la clase ArrayList<Vehiculo> y no añade ningún dato a su estructura, por lo que extiende únicamente su comportamiento, al añadir los métodos pedidos:

- registrarVehiculo.
- buscaVehiculo.
- precioAlquiler.
- mostrarTodosLosPrecios.

Se dispone del tester VehiculosTest y del esqueleto de Vehiculos para completar con la implementación de los métodos pedidos. En Vehiculos está la especificación informal de estos métodos.
Se pide:

- Realizar los métodos de Vehiculos que están sin hacer.

5. Clases *Peticion* y *Empresa*.

Material necesario

- Peticion.java
- PeticionTest.java
- Empresa.java
- EmpresaTest.java

Material a entregar

- Empresa.java

Las clases Peticion y Empresa modelan respectivamente la petición de un vehículo de alquiler por parte de un cliente y la empresa de alquiler de vehículos según lo explicado en la descripción general (sección 1).
Se dispone de la Peticion que se da hecha, de los testers PeticionTest y EmpresaTest y del esqueleto de Empresa para completar con la implementación de los métodos pedidos. En Empresa está la especificación informal de estos métodos.
Se pide realizar los métodos de Empresa que están sin hacer, que son:

- Constructor de Empresa.
- cargarFlota.
- alquilarVehiculos.
- mejorCliente.
- actualizar: método auxiliar para mejorCliente.