Universidad de Málaga

ETSI Informática

Patrones de diseño aplicados a un sistema de Alquiler de coches

Modelado y Diseño del Software (2024–25)

Daniil Gumeniuk
Diego Sicre Cortizo
Pablo Ortega Serapio
Angel Nicolás Escaño López
Francisco Javier Jordá Garay
Janine Bernadeth Olegario Laguit

Grupo 1.1

Diciembre 2024

Índice

Índice de figuras

Resumen

Esta práctica tiene como objetivo el diseño e implementación de un sistema de gestión de alquileres de coches para una empresa. El proyecto se centra en desarrollar las funcionalidades necesarias para gestionar el proceso de alquiler, las restricciones de negocio y las operaciones específicas requeridas asi como aplicar *Patrones de Diseño* presentados en el Tema 6.

Estructuramos el trabajo de tal forma que para cada uno de los apartados presentamos en esta memoria, se desarrolla y documenta un modelo UML hecho en *Visual Paradigm* detallado ajustándose a las necesidades de cada sección. También se justifican las posibles contradicciones o lagunas encontradas en el enunciado. Además, se explican las decisiones de diseño tomadas para la implementación del código en lenguaje *Java* respetando las restricciones impuestas por el enunciado.

Todas las implementaciones parten del mismo esqueleto que se presenta en la Introducción de la memoria, detallando en cada apartado las clases que se ven afectadas al implementar el *Patrón de Diseño* escogido (en caso que fuera necesario) y una explicación de como ese patrón enriquece el funcionamiento del sistema original.

1. Introducción

1.1. Diagrama de Diseño

Figura 1: Diagrama de Diseño del esqueleto

1.2. Cambios respecto a la implementación original

2. Ejercicio 1

Instrucción

Supongamos ahora que queremos definir una operación `numberOfRentalsWithDifferentOffices() : Integer` en la clase `Customer` que devuelve el número de alquileres web que ha hecho un cliente self donde la oficina de recogida y de entrega es diferente. En este momento del diseño del sistema todavía no sabemos qué estructura de datos utilizaremos para guardar los alquileres que ha hecho/hace un cliente.

2.1. Patrón de Diseño utilizado

2.2. Efectos sobre el Diagrama de Diseño

2.3. Implementación de *numberOfRentalsWithDifferentOffices() : Integer*

```
introducir codigo aqui
```

3. Ejercicio 2

Instrucción

A menudo, los coches que la empresa de alquiler de coches pone a disposición de sus clientes se tienen que poner fuera de servicio (por reparaciones, para pasar la ITV, etc.). Queremos representar esta situación en nuestro sistema para que cuando los coches estén fuera de servicio no puedan ser alquilados aunque registraremos, si hay, un coche que lo sustituye. Es por eso que nuestro sistema tendrá que proporcionar dos funcionalidades. La funcionalidad (1) poner un coche fuera de servicio (si ya está fuera de servicio o si está en servicio pero es sustituto de algún coche que está fuera de servicio, la funcionalidad no tendrá ningún efecto).

Esta funcionalidad pondrá el coche fuera de servicio y registrará la fecha hasta la cual estará fuera de servicio y, si hay, buscará y registrará también un coche sustituto (será cualquier coche del mismo modelo del coche que se pone fuera de servicio que esté asignado a la misma oficina y que esté en servicio).

La funcionalidad (2) pone un coche que estaba fuera de servicio en servicio.

En concreto, **nos centraremos en implementar la funcionalidad (1)** añadiendo la operación `takeOutOfService(backToService : date)` de la clase `Car`. No hace falta implementar la funcionalidad (2).

3.1. Patrón de Diseño utilizado

3.2. Efectos sobre el Diagrama de Diseño

3.3. Implementación de *takeOutOfService*

```
introducir codigo aqui
```

4. Ejercicio 3

Instrucción

Cuando los clientes de la empresa de alquiler de coches alquilan un coche, el sistema que estamos construyendo tiene que proporcionarles el precio del alquiler. Este precio lo calcula la operación `getPrice() : Integer` de la siguiente forma: El precio base será el precio del modelo del vehículo por día.

```
(pricePerDay) * [endDate - startDate]
```

Además, la empresa de alquiler de coches puede añadir al cálculo de precios la posibilidad de hacer promociones que implican descuentos de estos precios. Inicialmente, la empresa ofrecerá dos tipos de promociones: por cantidad y por porcentaje.

- **Promoción por cantidad:** permitirá decrementar el precio del alquiler en la cantidad indicada en la promoción.
- **Promoción por porcentaje:** decrementará el precio del alquiler en el porcentaje indicado en la promoción. Las promociones se asignan a los alquileres en el momento de su creación.

Evidentemente, es posible que a algunos alquileres no se les aplique ninguna promoción. Las promociones que se asignan a los alquileres son determinadas por una política de la empresa que no impacta al diseño de nuestra operación (impactará a la operación que crea los alquileres).

Eso sí, la empresa quiere que mientras no se haga el pago del alquiler, si aparecen nuevas promociones, se apliquen a los alquileres siempre y cuando sean más favorables (no nos tenemos que preocupar tampoco de estos cambios, son gestionados por otras operaciones).

4.1. Patrón de Diseño utilizado

4.2. Efectos sobre el Diagrama de Diseño

4.3. Implementación de *getPrice() : Integer*

```
introducir codigo aqui
```