TRABAJO PRÁCTICO FINAL

Programación Orientada a Objetos II

Integrantes:

● Lucas Paolo (Email: lucaspaolo002@gmail.com).

● Fernando Romero (Email: agustinfernando2000@hotmail.com).

● Mauricio Velazquez (Email: mauriciovelazquez211@gmail.com).

Decisiones de diseño:

- Al momento de realizar el gps, este lo podríamos haber implementado como si fuera una interfaz, pero ya que solo nos devolvía la zona en la cual se encontraba, hicimos que simplemente la AppUsuario posea esta zona.
- Incorporamos la clase Reloj que devuelve la hora actual para poder verificar los horarios de los estacionamientos tanto para iniciar como para finalizar en el test.
- En el patrón Observer, independientemente del evento que suceda, este se encarga de notificar a todas las entidades suscriptas.
- A la hora de realizar una compra, ya sea CompraPuntual o RecargaCelular se le pasa por parámetro una patente y luego el Sem es el encargado de asociar una patente a un usuario para poder realizar la recarga.

Patrones utilizados:

Patrón Observer:

Aplicamos este patrón para que los interesados en recibir notificaciones de inicio, fin de estacionamiento o recarga de saldo puedan suscribirse/desuscribirse del sistema cuando quieran.

Los roles entonces son:

- La clase **SEM** como el **Observable Concreto**.
- La clase **AppUsuario** con el rol de **Contexto**.
- La interfaz **INotificador** con el rol de **Observable**.
- La interfaz **ISuscriptor** con el rol de **Observer**.
- La clase **Suscriptor** con el rol de **ObservadorConcreto**.

Patrón State:

Aplicamos este patrón porque el ModoApp envía mensajes automáticos para iniciar y finalizar estacionamientos que dependen de si el usuario está estacionado o no.

Los roles entonces son:

- La clase **AppUsuario** con el rol de **Contexto**.
- La interfaz **EstadoEstacionamiento** con el rol de **State**.
- Las clases **EstacionamientoIniciado**, y **EstacionamientoSinIniciar** con el rol de **Estados Concretos** que implementan la interfaz EstadoEstacionamiento.

Patrón Strategy:

Aplicamos este patrón porque la AppUsuario contiene dos formas de iniciar y finalizar un estacionamiento (Manual/Automático) y el usuario de la misma puede elegir cual quiere.

Los roles entonces son:

- La **AppUsuario** con el rol de **Contexto**.
- La interfaz **ModoApp** con el rol de **Strategy**.
- Las clases **ModoAutomático** y **ModoManual** con el rol de **Estrategias Concretas** que implementan la interfaz ModoApp.

Como cada estrategia es independiente una de otra, se puede agregar en cualquier momento otro criterio.

Patrón Template Method:

Aplicamos este patrón ya que hay dos formas de obtener un estacionamiento que son tanto por app como por compra puntual.

Los roles entonces son:

- El **Estacionamiento** con el rol de **Clase Abstracta**.
- Las clases **EstacionamientoApp** y **EstacionamientoCompraPuntual** con el rol de **Clases Concretas**.

Patrón Template Method:

Aplicamos este patrón ya que hay dos formatos de compra y al necesitar registrarlos, se dividen en CompraPuntual y RecargaCelular

Los roles entonces son:

- La **Compra** con el rol de **Clase Abstracta**.
- Las clases **CompraPuntual** y **RecargaCelular** con el rol de **Clases Concretas**.