Trabajo Final
Programación con objetos II

Integrantes:

- Alvarez, Lucas (lucas.alvarez@alu.unq.edu.ar)
- Bender Defago, Gonzalo (gonzabender@gmail.com)
- Cervantes, Ignacio (ignacio.m.cervantes@gmail.com)

Profesores:

- Butti, Matias
- Cano, Diego
- Torres, Diego

Diseño:

Para el diseño se eligió utilizar la clase que representa el SEM como centro del modelo. Esta se encarga de llevar la información sobre todas las diferentes zonas que se encuentran en el sistema, sus estacionamientos, los usuarios y sus infracciones.

Las Zonas de Estacionamientos a su vez son los encargados de manejar los estacionamientos en esa zona, manteniendo una lista diferenciando los que están vigentes y lo que no. Además, cuenta con una lista con los respectivos puntos de venta.

Los Puntos de Venta son los encargados de gestionar las acciones del cliente cuando este no utiliza la App, permitiéndole cargar crédito e iniciar un estacionamiento (y a su vez informarlo al SEM).

Cada zona tiene también un Inspector asignado que se encarga de confirmar si un vehículo tiene un estacionamiento vigente y de no tenerlo le genera una infracción.

Los estacionamientos proveen información sobre el vehículo y el momento tanto de inicio como de fin del mismo. A su vez están clasificados en dos subclases según como fueron gestionados:

- EstacionamientoApp, cuando se realiza mediante el celular del cliente.

- EstacionamientoPuntual, cuando se le da inicio en un punto de venta.

Mediante la App el usuario puede realizar gestiones como cargar crédito y consultar su saldo, iniciar y finalizar un estacionamiento. Sumado a esto el usuario puede decidir si iniciar manualmente el estacionamiento o si lo hace la aplicación de manera automática y para esto la aplicación cuenta con los estados Automático y Manual de manera que cada una mantiene sus métodos que le permiten actuar diferente.

La App también cuenta con un sensor de movimiento que mediante un Observer determina (cuando este seteada en automático) si el usuario se encuentra manejando o no.

Por último, cada transacción que se realiza queda registrada mediante la clase Compra y se almacena en un registro de compras en el SEM.

Decisiones:

- Inicialmente se pensó implementar una interfaz Cobrable que unificara a las clases App y PuntoDeVenta, pero se decidió que no era necesaria.
- El modo manual y automático de la aplicación en un principio fueron subclases de la aplicación y luego se resolvió incorporándolo con un State.
- Se discutió si era necesaria la clase ZonaSEM, ya que el mismo SEM podría haber manejado la información, pero optamos por mantenerla porque tenía una responsabilidad en el diseño.

- Se decidió que SEM implemente la interfaz Observable para que distintas entidades sean alertadas sobre cambios en el sistema.

Patrones:

Usamos los siguientes patrones de diseño en la implementación del modelo:

El patrón Observer fue utilizado tanto para las entidades que podrían querer ser notificadas de cambios en el SEM como para detectar mediante la App si un vehículo se encontraba en movimiento o estacionado.

El patrón State se utiliza en la implementación de la App, ya que permite que al elegir si se la quiere usar en Modo Automático o Manual, sean estos quienes decidan el comportamiento.

El patrón Adapter fue utilizado para que las entidades que quieren ser informadas de los cambios en el SEM sean polimórficas y puedan comunicarse.

Reentrega:

- El estado de desplazamiento fue modelado con un state.
- Múltiples métodos de estado app fueron delegados.
- Las fechas fueron modeladas con la clase Calendar y las horas con la clase LocalTime.
- RecargaCelular ahora conoce al celular como objeto.
- Las clases fueron agrupadas en paquetes específicos.

- Cálculo del saldo fue delegado a la clase Celular.
- El fin de horario de estacionamiento ahora lo decide el usuario mediante la app.
- Finalizar todos los estacionamientos ahora tambien descuenta el saldo de todos los celulares que estaban estacionados.
- Todos los strings son comparados con el mensaje equals.
- Aplicamos test doubles a los test.
- Se mejoró el patron adapter para las entidades interesadas.
- Las franjas horarias del sistema fueron modeladas mediante el patrón State.