Trabajo Práctico: Sistema De Estacionamiento Medido

Integrantes del grupo:

Lautaro Galvez Monge: lautaroarielgalvezmonge@gmail.com

Nicolas Vaccaro: nicolasvaccaro48@yahoo.com.ar

Lucas Sanguinetti: lucas.e.sanguinetti@gmail.com

Materia y ciclo: Programación con Objetos 2 primer cuatrimestre 2024

Profesores: Diego Cano, Matias Butti, Diego Torres.

Decisiones de Diseño :

Decisiones Orientadas A Datos:

- Los datos del tipo Hora y Fecha en lugar de ser usados como datos de su propio tipo, se decidió reemplazarlos por números de entre 0 y 23 y Strings de formato "DD/MM/AAAA" respectivamente, esto se hizo de esta manera para una major facilidad a la hora del manejo y seteo de estos, ya que el manejo de estos tipos es más familiar para todos los miembros del equipo y permite trabajar con la parte importante del trabajo, que es diseñar el sistema, sin trabarse con el uso de datos que no han sido trabajados demasiado.
- En la clase 'ZonaEstacionamiento' para conocer su inspector, ya que existe uno solo por zona, decidimos que solo conozca el id de este mismo guardando en una variable de tipo int. Ya que este id es el que identifica al inspector y se asume que es único entre ellos. Se asume que no existe en ningún momento un inspector sin zona ni una zona sin inspector.
- En el 'SEM', posee como variable un int llamado "contadorDeTicekts". Esta variable tiene la función de llevar registro de la cantidad de tickets que se emiten en el sistema y poder usarlos como identificador de los tickets.
- Las patentes de los autos se decidió que sean String con el siguiente formato: "LL-NNN-LL", siendo l una letra del alfabeto u n un numero de un dígito, para todas las patentes que se manejan dentro de nuestro modelo. Cada patente se asume que es única y que respeta el formato.
- En la clase 'AppUsuario' existen dos booleanos que sirven como bandera. El bool 'vigente' es para obligar al usuario a tener solo un estacionamiento a la vez. En el otro bool 'sensorMovimiento' sirve para saber si el sensor está prendido, ya que los mensajes walking y driving solo deben ser usados si el sensor está prendido.
- Manejo de null, en algunos casos del trabajo, cuando se busca un objeto se usa el mensaje findFirst().orElse(Null), esto se hace así para tipar el objeto con el tipo que se requiere y no como un <Optional> , dentro del trabajo se plantean test que trabajen cuando se está y cuando no se encuentran algunos objetos. No debería de pasar en ningún momento que se trabaje con un null y no se tome una previsión de eso, como hacer short circuit o crear objetos nuevos/dummy que den los datos necesairos.

Decisiones De Clases:

- .Para registrar las compras se decidió guardar en una lista con el tipo 'Ticket' dentro del SEM. Para poder distinguirlos se crearon dos clases distintas (TicketEst y TicketSaldo) que implementan la interfaz Ticket para cumplir con el polimorfismo.
- Para los registros de estacionamientos se creó una clase llamada 'RegistroEst', que nos sirve para poder identificar que los estacionamientos estén vigentes dentro del SEM. Además para que se diferencien entre ellos existen 2 constructores, uno para el estacionamiento por punto de venta y el otro para el estacionamiento por app, donde el mismo registro tiene la particularidad de saber cuánto dura cada estacionamiento indicando en qué momento finalizan.

- Para registrar las cargas de crédito se decidió crear una clase 'Credito'. Además el crédito mismo es el que sabe responder cuanto saldo posee, esto para facilitar en caso de que se esté cargando saldo a un número que ya posee crédito,además facilita la búsqueda de saldo por número de teléfono en caso de que estos teléfonos recién hayan descargado la aplicación y no tengan su saldo registrado. Si bien se podría crear el crédito ni bien la aplicación es instalada, esto no se hace para evitar crear saldos que no van a ser consultados nunca.
- Por el tema de infracciones también se creó una clase llamada 'Infracción' donde esta tiene todo los requerimientos que pide el trabajo.
- Con el inspector se optó por crear directamente la AppInspector, y que esta misma tenga un id que le identifique. Esto fue hecho así porque crear la aplicación ya tiene el comportamiento deseado para el sistema, el inspector en sí es un usuario y por ende no debe de ser implementado.
- Dentro de la clase SEM, hay dos formas de enviar la notificación del observer, cuando inicia el estacionamiento y cuando termina, en el caso de terminar, es la zona de estacionamiento la que manda el mensaje detonador del allNotify(), sin embargo, en el inicio de estacionamiento, es el SEM el que notifica este cambio, ya que guardar un RegistroEst se hace la misma cantidad de veces que el estacionar(patente) de las zonas de estacionamiento.

Decisiones De Diseño es base a patrones:

Dentro de este trabajo nos encontramos con tres patrones, a detallar a continuación:

Observer:

- Dentro de la consigna se menciona que el SEM desea poder suscribir a las notificaciones de inicio y fin de estacionamiento a otras entidades que deseen ser notificadas de eso, para lograr esto se implementa este patrón, ya que ayuda a agregar, des-agregar y notificar a esas entidades de los inicios y finales de estacionamientos. los roles son los siguientes:
    - SujetoConcreto: SEM.
    - ObservadorConcreto: de momento no hay ninguno en concreto, pero se prevé que los haya, como por ejemplo un call center.
    - Que Detona El Notify: iniciar o terminar un estacionamiento.

Strategy:

- dentro de la aplicación nos encontramos con dos modos de operar, el manual, donde el cliente decide qué acción realizar, inicio/fin estacionamiento y el automático que decide en base a un sensor, dentro del trabajo esto fue pensado en primera instancia como un state, pero debido a dos factores esta corriente de pensamiento fue cambiada, el primero de los factores fue el hecho de que el cliente puede elegir cual modo está usando en todo momento, y el segundo factor fue el sensor de movimiento, que si bien puede ser pensado que en base a la acción de detener o activar el sensor el estado deba cambiar (apuntando a un patrón state), es el usuario el que elige activar o desactivar el sensor, implicando también que es él el que está decidiendo cambiar la forma en la que opera el teléfono.

Strategy:

- Contexto: AppUsuario.
- Strategy: Modo.
- MensajesStrategy = incioEstacionamiento(patente): y finEstacionamiento().
- FirstStrategy:Modo Automatico.
- SecondStrategy: Modo Manual.