DOCUMENTACIÓN TRABAJO INTEGRADOR

INTEGRANTES

- Nasr, Cristian: nasrcristian.un@gmail.com
- Villalba, Damian: damianalumno@gmail.com

DECISIÓN DE DISEÑO

Basamos nuestro diseño enfocado en el enunciado y centralizando los mensajes en el SEM para que este pueda delegar a sus colaboradores más importantes que son los gestores:
GestorCuenta: se encarga de almacenar las cuentas, permite darle informacion relacionada al saldo o buscar alguna cuenta al sistema.
GestorEstacionamiento: se encarga de almacenar los estacionamientos (puntuales o por app), permite iniciar, finalizar y consultar los estacionamientos del sistema entre otras cosas.
GestorRegistro: se encarga de almacenar los registros (de recarga o compras puntuales), permite guardar, eliminar y consultar los registros del sistema.
GestorInfracciones: se encarga de almacenar las infracciones, permite generar infracciones y consultar por ellas al sistema

Luego, encapsulamos las clases necesarias para cumplir con la función estricta, caso contrario derivarlo a la clase responsable. Además desarrollamos e implementamos patrones de diseños (comentados más abajo) que mantienen la lógica y el encapsulamiento, cumpliendo con lo solicitado.

DETALLES DE IMPLEMENTACIÓN

- Uso de Optional para un mejor manejo de objetos y validaciones
- Uso de Excepciones para un mejor manejo de flujo del programa y notificaciones

PATRONES DE DISEÑO UTILIZADOS

Observer*(SistemaCentral a Entidades):*
Utilización del patrón para que el SistemaCentral pueda suscribir y desuscribir a las Entidades, y a su vez, notificarle cambios como el inicio o fin de un estacionamiento y recarga de crédito exitosa. Esto permite una mayor versatilidad al programa donde futuras entidades tendrán la posibilidad de insertarse y trabajar directamente según la notificación que reciban del sistema simplemente implementando las acciones hacia ellas.

Roles:
- *ISistemaObservable: Sujeto*
- *IEntidad: Observador*
- *SistemaCentral: SujetoConcreto*

State*(AppCliente: ConEstacionamiento y SinEstacionamiento):*

La AppCliente cambia de estado de manera automática, controlando el comportamiento del inicio y el fin del estacionamiento dependiendo del estado actual (toda AppCliente inicia con el estado SinEstacionamiento).

Roles:

- *AppCliente: Contexto*
- *EstadoApp: Estado*
- *ConEstacionamiento: EstadoConcretoA*
- *SinEstacionamiento: EstadoConcretoB*

Strategy*(AsistenciaAlUsuario y ModoDeApp):*

- *AsistenciaAlUsuario:*
  El usuario determina si va ser notificado por la app ante cambios de movimiento dependiendo de la estrategia actual.

  Roles:
  - *AppCliente: Contexto*
  - *AsistenciaAlUsuario: Estrategia*
  - *AsistenciaActivada: EstrategiaConcretaA*
  - *AsistenciaDesactivada: EstrategiaConcretaB*

- *ModoDeApp:*
  El usuario determina si va a ser responsable el o la app en iniciar o finalizar sus estacionamientos según los cambios de movimientos.

  Roles:
  - *AppCliente: Contexto*
  - *ModoDeApp: Estrategia*
  - *ModoAutomatico: EstrategiaConcretaA*
  - *ModoManual: EstrategiaConcretaB*