A

INFORME – SISTEMA ESTACIONAMIENTO MEDIDO (SEM)

ALUMNOS:

Gómez Gabriel

García Axel

Torres Nicolas

Registro de compras de parkings puntuales

Para llevar a cabo el registro de parkings puntuales se desarrolló la clase Compra, con la subclase CompraDeEstacionamiento, que permite almacenar en el SEM cada una de las compras realizadas y que no se pierdan, por si se quiere llevar a cabo un análisis de lo recaudado, etc.

Registro de parkings

Para registrar los estacionamientos dentro del sistema se desarrollo la clase Parking, esta posee dos subclases, ParkingPuntual y PaerkingViaApp lo que permite tener almacenados a todos los parkings dados de alta.

Inicialización de un UserApp

Al iniciar un UserApp (aplicación de usuario) esta es inicializada sin estacionamientos vigentes, con las notificaciones activadas y en modo manual (el modo puede ser modificado a gusto del usuario, entre los modos manual y automático).

Recibe como argumento un numero de celular y una patente predeterminada que será usada en caso de estar en modo automático.

Detalles de implementación:

Inicio de parking como usuario de app: El usuario puede elegir entre dos modos de uso, manual y automático.

Para el modo de uso **manual** el usuario tiene por defecto activadas las notificaciones, las cuales puede desactivar en cualquier momento, mientras este activado el usuario recibirá notificaciones que le recuerdan que posiblemente debe iniciar el estacionamiento, si esto es lo que el usuario desea hará manualmente un llamado al método: **iniciarParking(String patente)** el cual iniciara el estacionamiento con la patente pasada como argumento, este método le delegara el trabajo al estado del parking por lo que si hay un parking vigente el usuario no pueda iniciar más de 1 estacionamiento.

Cuando el modo de uso de la app es **automático** la aplicación mediante el sensor de movimiento detecta los cambios de desplazamiento del usuario y con ello iniciara el estacionamiento. En este caso, si el sensor de

movimiento detecta un cambio de desplazamiento de conduciendo a caminando y se encuentra en alguna zona de jurisdicción del sistema SEM, iniciara el estacionamiento automáticamente con la patente predeterminada que brindo el usuario al inicializar la app (se considera que la patente puede cambiar al cambiar de vehículo o que el usuario puede tener más de un vehículo por lo tanto la patente predeterminada puede ser cambiada por el usuario).

Finalización de parking como usuario de app: El usuario puede elegir entre dos modos de uso, manual y automático.

Para el modo de uso **manual** el usuario tiene por defecto activadas las notificaciones, las cuales puede desactivar en cualquier momento, mientras este activado el usuario recibirá notificaciones que le recuerdan que posiblemente debe finalizar el estacionamiento, si esto es lo que el usuario desea hará manualmente un llamado al método: **finalizarParking()** el cual finalizara el estacionamiento actual vigente, este método le delegara el trabajo al estado del parking por lo que si no hay un parking vigente no sucede nada, de lo contrario finalizara el parking vigente.

Cuando el modo de uso de la app es **automático** la aplicación mediante el sensor de movimiento detecta los cambios de desplazamiento del usuario y con ello iniciara el estacionamiento. En este caso, si el sensor de movimiento detecta un cambio de desplazamiento de caminando a conduciendo y se encuentra en alguna zona de jurisdicción del sistema SEM, finalizara el estacionamiento automáticamente.

Estados en UserApp

Métodos driving() y walking(): EstadoDeDesplazamiento

UserApp implementa la interfaz MovementSensor, dentro de estos métodos se delega al estadoDesplazamiento del UserApp, ya que, se considera que el sensor de movimiento invoca el método constantemente según el tipo de desplazamiento que este detectando, de esta manera, mientras este detectando que el desplazamiento es conduciendo si el estado es conduciendo no hacer nada hasta detectar otro tipo de desplazamiento y viceversa para el estado caminando. Una vez se detecta que cambio el estado de desplazamiento se delega el trabajo al modo de operación de la aplicación.

Modos de operación

Modo manual:

Cuando el estado de desplazamiento delega el trabajo al modo de operación, si este es manual, chequeara con el estado del parking si esta vigente un estacionamiento o no, si un parking no esta vigente y la aplicación tiene las notificaciones activadas se le enviara al usuario una notificación dependiendo el estado del sensor recordándole que debe iniciar o finalizar el parking. De lo contrario, si ya se tiene un parking vigente, no se realizan acciones.

Modo automático:

Cuando el estado de desplazamiento delega el trabajo al modo de operación, si este es automático, chequeara con el estado del parking si está vigente (porque quizás se inicia un parking de manera manual y luego se cambia el modo a automático), si no hay parking vigente y se detecta un cambio de desplazamiento de conduciendo a caminando se iniciara un parking de forma automática, si hay un parking vigente y se detecta un cambio de desplazamiento de caminando a conduciendo se finalizara el parking de forma automática.

EstadoParking

La clase abstracta EstadoParking define los métodos abstractos:

- iniciarParking(UserApp app, String patente);
- finalizarParking(UserApp app);

y define los siguientes métodos sin comportamiento:

- notificarInicioParkingPosible(UserApp app);
- notificarFinParkingPosible(UserApp app);

EstadoParkingNoVigente:

Esta clase que extiende de la clase abstracta EstadoParking define iniciarParking con los algoritmos y la lógica del tipo de negocio, el método finalizarParking lo define como un mensaje de alerta indicando que no se puede finalizar un estacionamiento que no está vigente, notificarInicioParkingPosible le da comportamiento para notificar al usuario y notificarFinParkingPosible no lo define, usa el de la clase abstracta.

EstadoParkingVigente:

Esta clase que extiende de la clase abstracta EstadoParking define finalizarParking con los algoritmos y la lógica del tipo de negocio, el método iniciarParking lo define como un mensaje de alerta indicando que no se puede iniciar un estacionamiento ya que se encuentra uno vigente notificarFinParkingPosible le da comportamiento para notificar al usuario y notificarInicioParkingPosible no lo define, usa el de la clase abstracta.

PATRONES DE DISEÑO UTILIZADOS

Para el desarrollo del trabajo practico se utilizaron tres patrones de diseños vistos en la cursada, estos fueron:

Dos state, uno para definir el estado de desplazamiento del usuario que está usando la aplicación y otro para definir el estado de un usuario, si se encuentra con un estacionamiento vigente o no.

Los roles de cada una de las clases en el desarrollo del estado son:

- UserApp es el “Context”
- EstadoDeParking, EstadoDeDesplazamiento son los state.
- EstadoDeParkingVigente y EstadoDeParkingNoVigente son los “concrete state” de EstadoDeParking
- EstadoConduciendo y EstadoCaminando son los “concrete state” de EstadoDeDesplazamiento.

Un strategy para elegir el modo de operación del UserApp, las estrategias son ModoManual y ModoAutomatico.

Los roles de cada una de las clases en el desarrollo del strategy son:

- UserApp es el “Context”
- ModoDeOperacion es el “strategy”
- ModoManual y ModoAutomatico son los “concrete strategy”

Finalmente, un Observer, para notificar sobre los inicios de parkings, finalizaciones de parkings y carga de saldo a usuarios.

La clase SEMSystem es el “Observable” y los “Observer” son las entidades que implementan la interfaz NotifyEntidad.