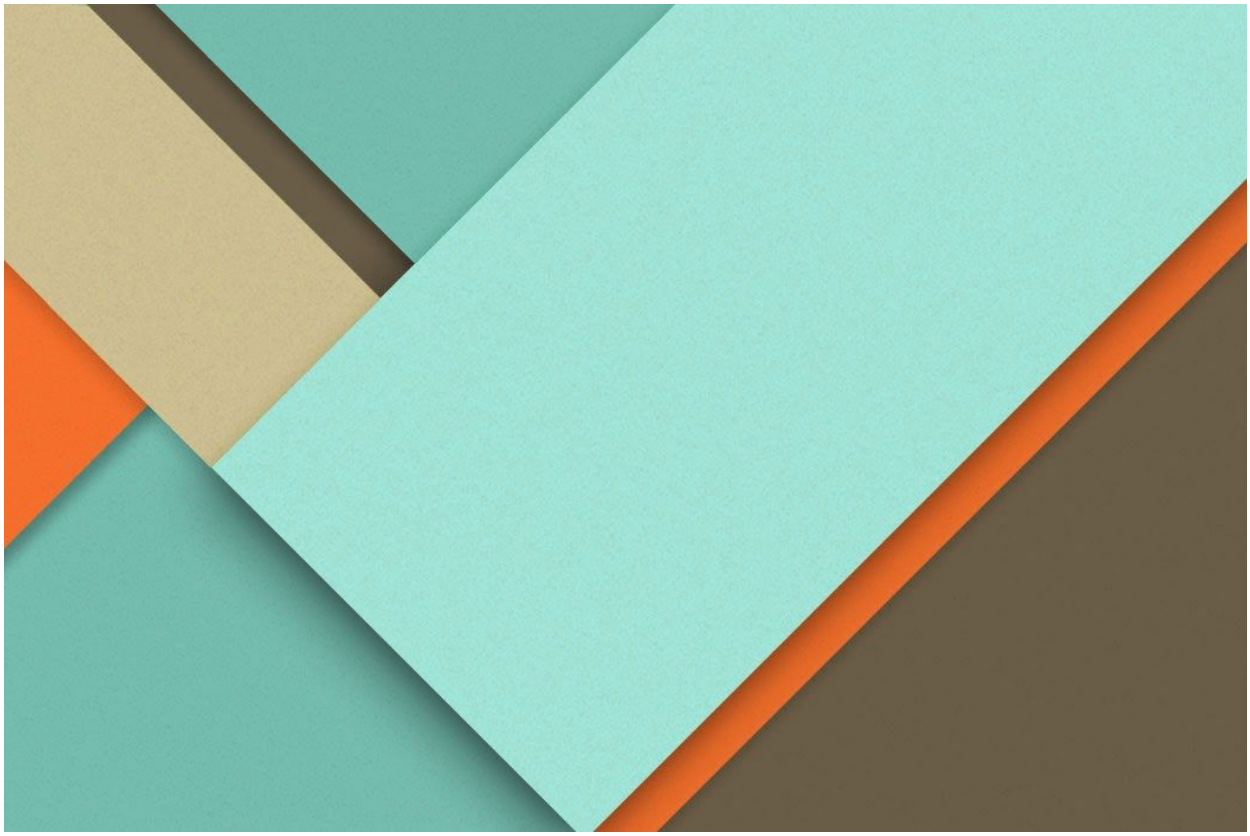


Sistema de Estacionamiento Medido

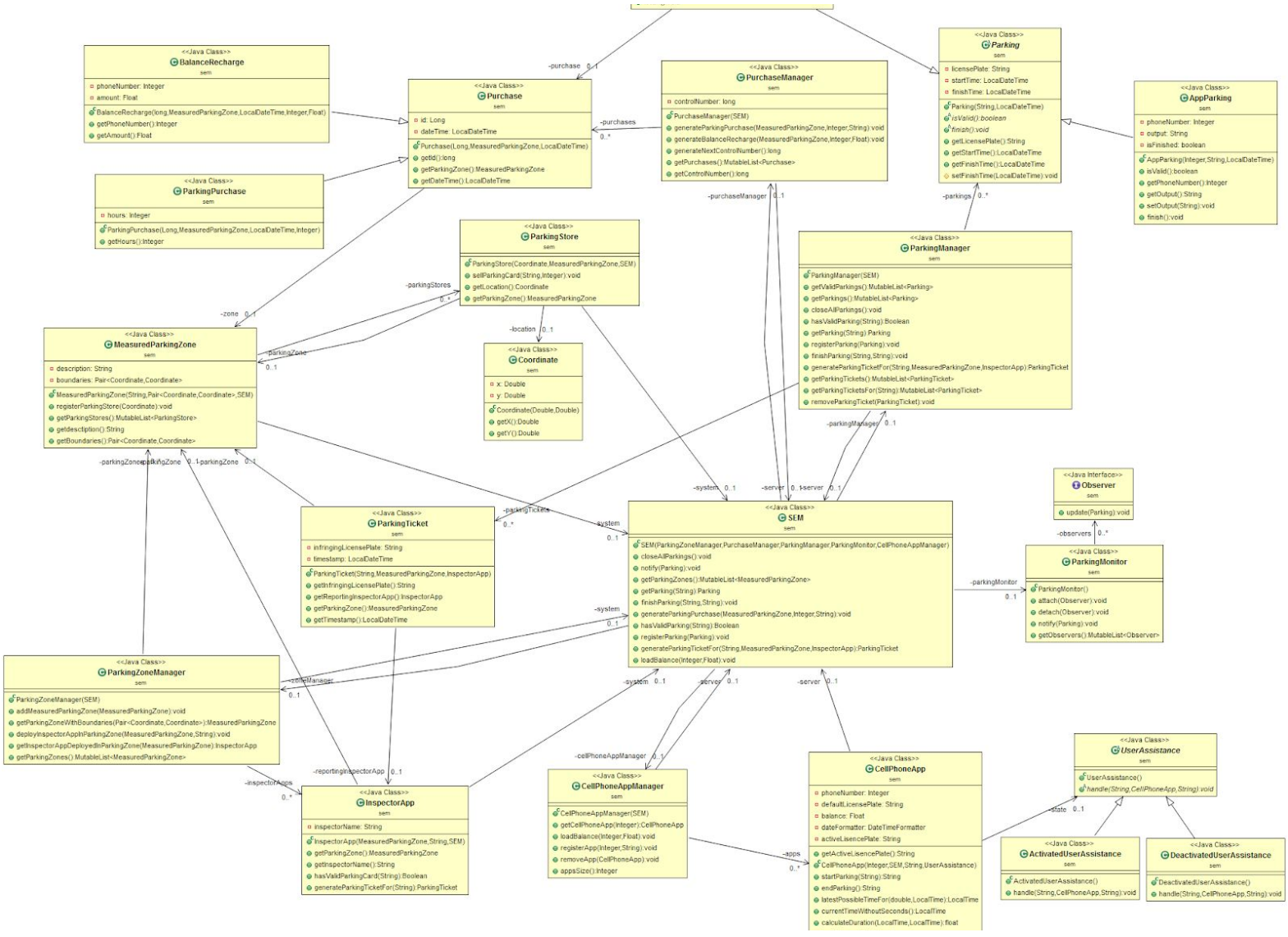
12.11.2020

Fernandez, Matias. fnandez.matias@gmail.com

Gattone, María de los Angeles. mariadelosangelesgattone@gmail.com

Martinez Lopez, Lautaro Gaston. lautaro.ml@gmail.com

UML



Decisiones de Diseño:

Una decisión que tomamos es manejar las colecciones con su propio manager para quitar responsabilidad al server.

Por ejemplo, en el caso de los estacionamientos generamos una clase que se encarga de administrar todo acerca de ellos y dentro de la misma hay una colección donde se guardan los estacionamientos generados, para de esta manera no sobrecargar el sistema.

Patrones de Diseño:

I. Observer para el ParkingMonitor.

Observer es un patrón de diseño de comportamiento que te permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando.

En este caso se utilizó un Observer para que otras entidades o sistemas sean notificadas cuando se inicia o se finaliza un estacionamiento, para así tener un control del número de espacios o para notificar a los usuarios cuando inician estacionamientos con poco saldo.

Al generarse un evento en el sistema estas entidades o sistemas, serán alterados o notificados.

II. State para userAssistance.

State es un patrón de diseño de comportamiento que permite a un objeto alterar su comportamiento cuando su estado interno cambia. Parece como si el objeto cambiará su clase.

Utilizamos el state en este caso, por que, al activarse la asistencia al usuario el comportamiento del sistema cambia e indica que tiene que cobrar y activar el estacionamiento cuando detecte que el usuario está caminando, además de enviar un alerta a la app del celular. Y al desactivarse, el sistema no realiza cambios automáticamente, como activar y cobrar el estacionamiento, simplemente manda una notificación en la app del celular.

III. Facade para el SEM.

Facade simplifica la complejidad de un sistema mediante una interfaz más sencilla. Mejora el acceso a nuestro sistema logrando que otros sistemas o subsistemas usen un punto de acceso en común que reduce la complejidad, minimizando las interacciones y dependencias.

Utilizamos facade para centralizar el uso de los distintos managers pero manteniendo sus responsabilidades en sus respectivas clases.

Por ejemplo, para registrar los estacionamiento el SEM simplemente va a utilizar una clase que contiene todas las responsabilidades para generar un estacionamiento adecuadamente.

Los roles según la definición de Gamma et. al.

I. Observer.

La clase ParkingMonitor es el observer. Los subsistemas y entidades que utilicen ParkingMonitor, son los subscriptores.

II. State.

En este caso, la clase CellPhoneApp es la que utiliza el State. Y UserAssistance cumple con el rol de Abstract State, mientras que ActivatedUserAssistance y DeactivatesUserAssistance cumplen con el rol de Concrete State

III. Facade.

La clase SEM es el Facade y las clases ParkingManager, PurchaseManager, ParkingZoneManager y CellPhoneAppManager son subsistemas que se encargan de manejar las comprar, los estaciomientos y las app de celular.