PO2:
Sistema de Estacionamiento Medido

Integrantes grupo 3:

Fabricio Marote - fabrii.cai93@gmail.com
Zarate Cristian Gonzalo - nofuture.gonza@gmail.com
Manuel Michellon - manuel.michellon@gmail.com

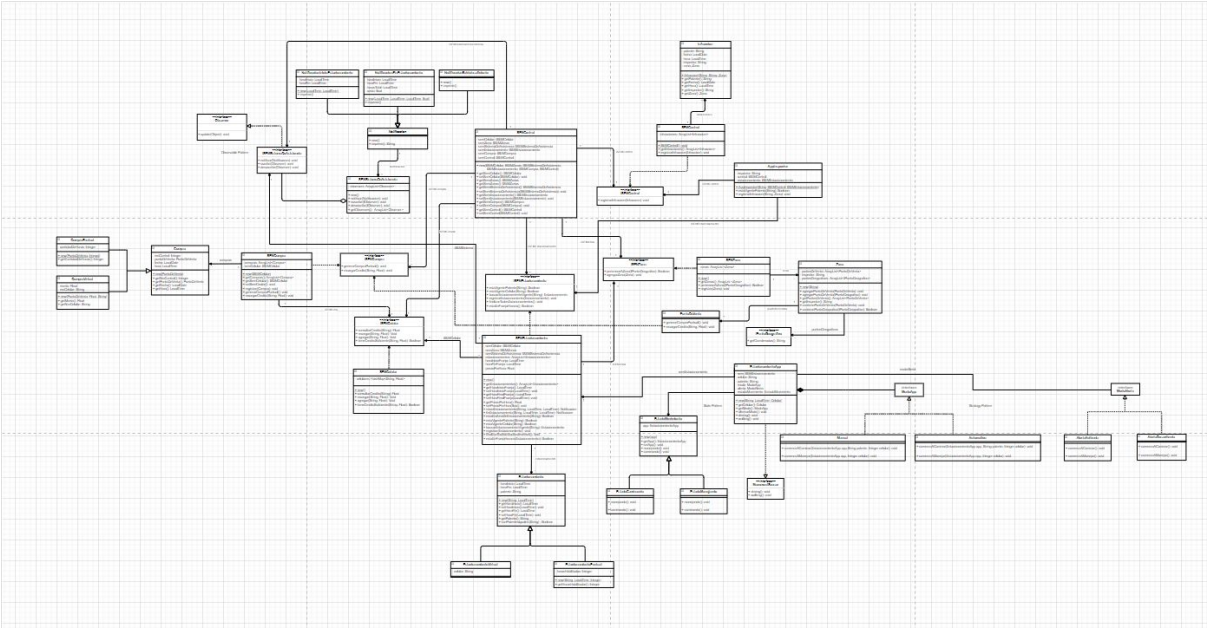
Java Compiler:

JDK-14.0.2

External Jars:

Se utilizaron los archivos externos que pasaron al Slack para poder utilizar Mockito.

Diseño



Al empezar el TP Final diseñamos e implementamos SEM como una clase que administraba todas las funciones del proyecto, pero luego refactorizamos el diseño pensando en que no es necesario que SEM sea una clase que haga todo y sobrecargarla de funcionalidades.
Así fue como decidimos pensar en el proyecto como si tuviéramos diferentes “áreas” o “universos” independientes con sus respectivas funciones, abstrayendonos de SEM, esto permite que cualquier otra clase o aplicación que necesite implementar estos universos también podría hacerlo mediante las interfaces que brindamos en cada uno de ellos.
También nos permitió separar más fácil las tareas y trabajar de forma ordenada.
Cada área o universo de nuestro proyecto fue dividido en un paquete dentro del eclipse.

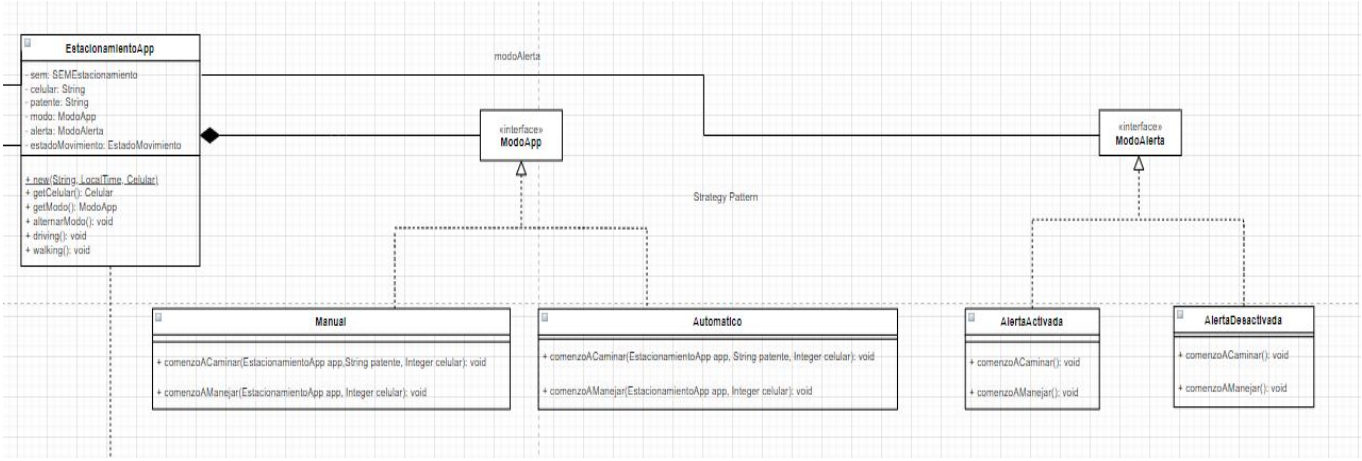
Los universos son:

- ISEMEstacionamiento: encargado de administrar y operar todo lo relacionado con estacionamientos y sus app.
- ISEMCompra: encargado de gestionar todas las compras.
- ISEMControl: encargado de las infracciones y lo relacionado con ellas
- ISEMZona: encargado de las zonas y puntos de ventas que se encuentran en ellas.

Patrones de Diseño

Strategy:

La App de estacionamiento implementa el siguiente patrón cuando hablamos de Alertas y Modos de utilizarla.



Se utiliza un Strategy ya que tenemos diferentes maneras de utilizar la app y esto cambia cuando el usuario lo desee

- Modo Automático
- Modo Manual

El contexto es EstacionamientoApp, el strategy es ModoApp y las clases concretas del strategy son Manual y Automatico.

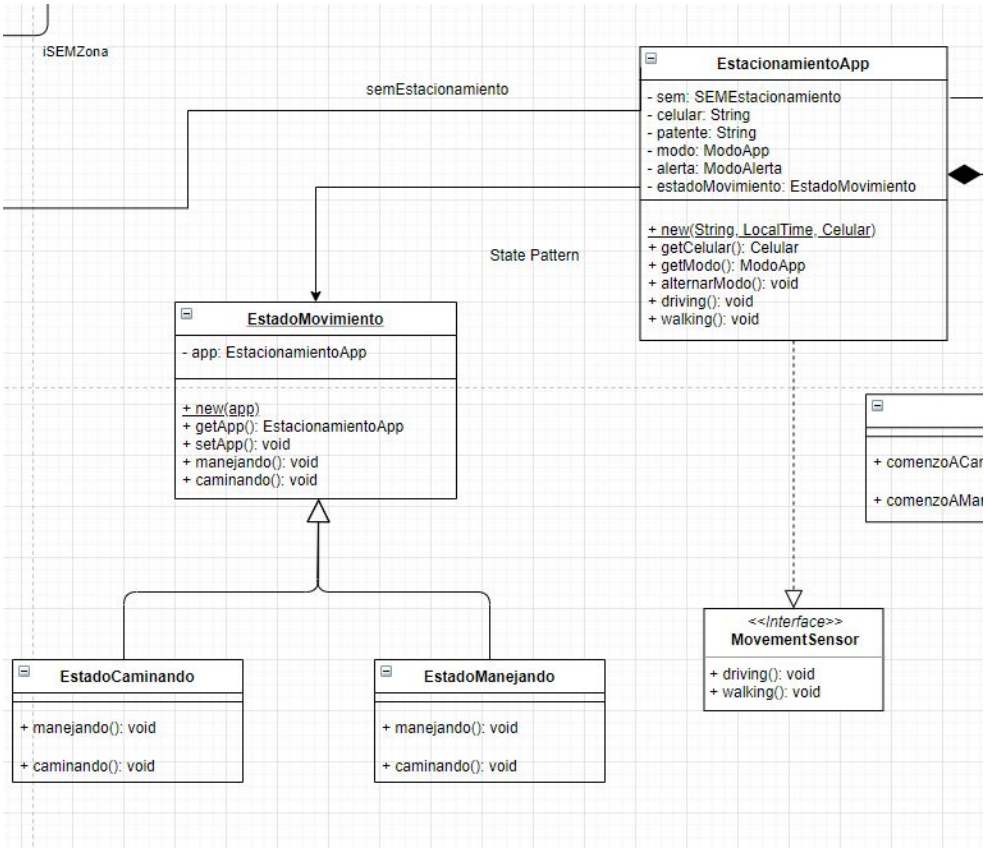
Y lo mismo ocurre con las alertas

- Alerta Activada
- Alerta Desactivada

El contexto es EstacionamientoApp, el strategy es ModoAlerta y las clases concretas del strategy son AlertaActivada y AlertaDesactivada.

State:

La App de estacionamiento cambia de estado dependiendo de lo que el MovementSensor le envié, los posibles mensajes son walking() y draving().

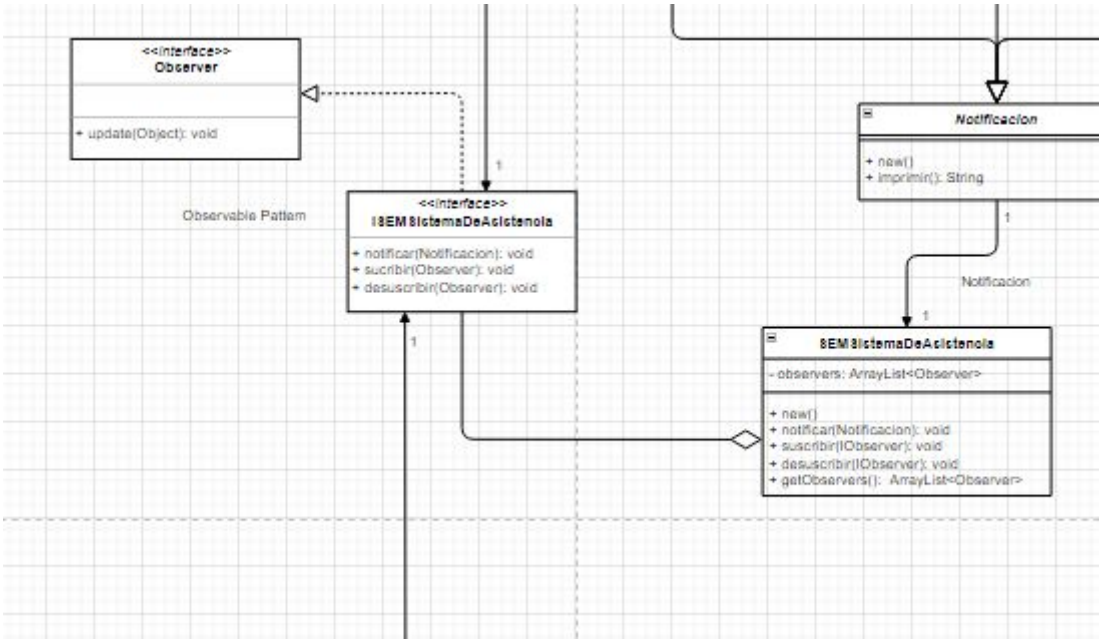


Dependiendo en qué estado se encuentre la aplicación, se va a compartir de una manera o otro según el mensaje que MovementSensor le envié.

El contexto es EstacionamientoApp, el state es EstadoMovimiento y las clases concretas del state son EstadoCaminando y EstadoManejando.

Observer:

Se implementó un sistema de asistencia al usuario. El cual mediante el mensaje notificar envía mensaje a una colección de observers, para notificarlos mediante mensajes preseleccionados.

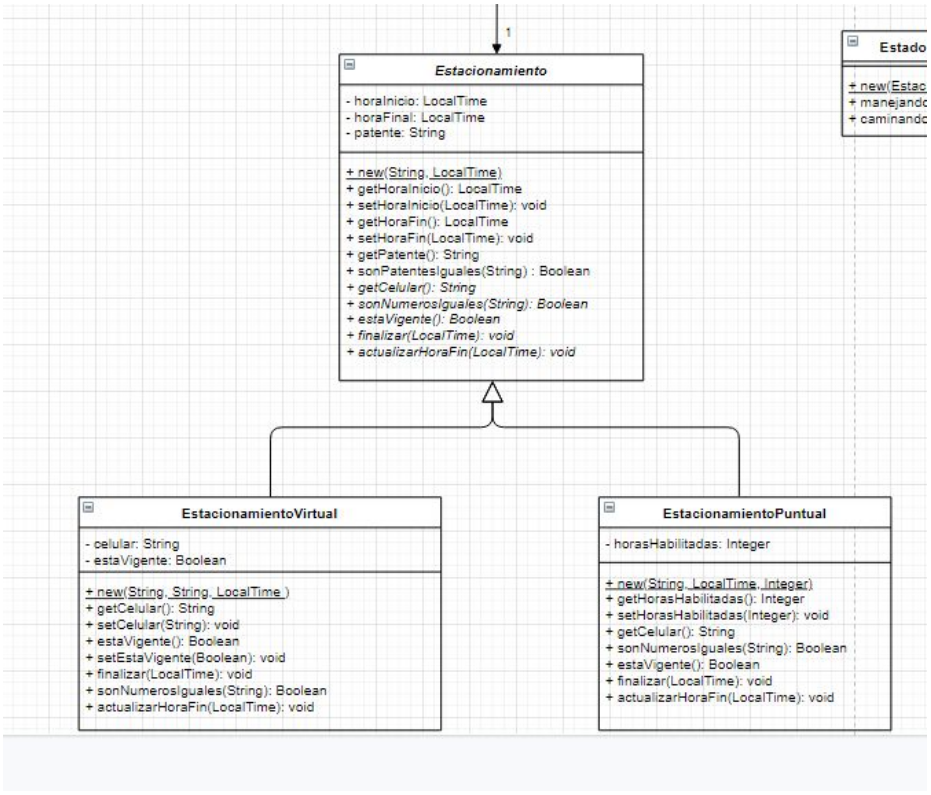


Los observers concretos quedaron a disposición de quien quiera implementar la interfaz y subscribirse, el observable concreto o concrete subject es el SEMSistemaDeAsistencia con su correspondiente interfaz. Mediante el mensaje update nos aseguramos que todos los observers reciben las notificaciones.

Template Method:

Se aplicó template method en dos ocasiones:
Al crear la clase estacionamiento con sus hijos y al crear la clase Notificacion y sus hijos.

Estacionamiento:
La clase abstracta es Estacionamiento y sus hijos/clases concretas son Estacionamiento Virtual y Estacionamiento Puntual.
Contamos con 5 métodos abstractos (operaciones primitivas) que son implementadas en las clases concretas. Contamos con dos hook methods finalizar y actualizarHoraFin.


1
Estacionamiento
- horaInicio: LocalTime
- horaFinal: LocalTime
- patente: String
+ new(String, LocalTime)
+ getHoraInicio(): LocalTime
+ setHoraInicio(LocalTime): void
+ getHoraFin(): LocalTime
+ setHoraFin(LocalTime): void
+ getPatente(): String
+ sonPatentesIguales(String) : Boolean
+ getCelular(): String
+ sonNumerosIguales(String): Boolean
+ estaVigente(): Boolean
+ finalizar(LocalTime): void
+ actualizarHoraFin(LocalTime): void
Estado
+ new(Estac
+ manejando
+ caminando
EstacionamientoVirtual
- celular: String
- estaVigente: Boolean
+ new(String, String, LocalTime)
+ getCelular(): String
+ setCelular(String): void
+ estaVigente(): Boolean
+ setEstaVigente(Boolean): void
+ finalizar(LocalTime): void
+ sonNumerosIguales(String): Boolean
+ actualizarHoraFin(LocalTime): void
EstacionamientoPuntual
- horasHabilitadas: Integer
+ new(String, LocalTime, Integer)
+ getHorasHabilitadas(): Integer
+ setHorasHabilitadas(Integer): void
+ getCelular(): String
+ sonNumerosIguales(String): Boolean
+ estaVigente(): Boolean
+ finalizar(LocalTime): void
+ actualizarHoraFin(LocalTime): void