Trabajo Práctico Integrador

Grupo 7

Integrantes

- David Ignacio Duarte - davidignacio20@outlook.com
- Juan Pablo Díaz - juampi00.est@gmail.com
- Leandro Tittarelli - leandrotittarelli@gmail.com

Se decidió utilizar el patrón de diseño Singleton sobre la clase SEM, debido a que solo se utiliza una instancia de la misma en toda su ejecución. Si bien no es completamente necesaria su implementación, vimos su uso acorde a esta situación.

Además del uso del Singleton, se implementó el patrón Observer debido a la necesidad de tener la flexibilidad de tener que notificar desde el SEM a otras entidades del manejo de los estacionamientos, aunque no aparezcan en el modelo, siendo nada más que una abstracción.

También se aplicó el patrón Strategy para los modos manual y automático de la clase AppUsuario. Ya que nos permitió facilitar el algoritmo a ejecutarse del manejo de un estacionamiento dependiendo del modo de desplazamiento del momento.

Se generó la clase SistemaDeAsociaciones que mapea el número de teléfono de cada usuario de una app, con su patente y su saldo. Esta decisión fue para quitarle responsabilidad al SEM y contar con un sistema más específico que pueda tener un pleno control sobre estos maps.

Por último, se implementó un método estático “clearSEM” dentro de la clase SEM, que setea al mismo con “null”, ya que gran parte de los tests que requerían usar esta clase rompían: los métodos “setUp()” de los archivos de tests dejan, en muchas ocaciones, una instancia del SEM distinta de la inicial (null).