Programación con objetos II

Trabajo final: Sistema de Estacionamiento Medido

Integrantes:

Bruno Agustín Chacana - chacanabruno@gmail.com

Benjamín Domínguez - benjadominguezf@hotmail.com

Francisco Salig - franciscosalig@gmail.com

Decisiones de diseño respecto al Sistema de Estacionamiento Medido.

El SEM se encuentra abierto para registraciones las 24 hs. Sin embargo, tiene una franja horaria donde se cobran los estacionamientos (7 a 20). Decidimos que los usuarios tengan la posibilidad de registrar estacionamientos en cualquier momento y solo se les cobrara las horas respectivas que estén dentro del horario laboral. Esta fue una decisión de diseño para solventar casos respecto a la App en modo automático.

Es decir, un usuario decide tener su aplicación en modo automático para no hacerse cargo de iniciar y finalizar manualmente. El usuario estaciona 5am y vuelve 10am. El sistema registrara su estacionamiento y a la hora de finalizarlo solo le cobrara las horas respectivas a la franja horaria (3 horas en este caso).

Es elección del usuario registrar su auto en el sistema en **horario no cobrable**. En caso de que no lo registre y se encuentre dentro la franja horaria laboral, el inspector puede realizarle una infracción.

Explicación funcionamiento de cobro.

Tenemos la clase **<Rango Horario>** que determina un intervalo de tiempo, tiene una horaInicio y una horaFin.

Tenemos la clase **<CalculadorDeTarifa>** que tiene una lista de tarifas con los rangos posibles de estacionamientos que se puedan recibir.

Escenarios de Tarifas

-**TarifaDentroDeHorarioLaboral**:

El estacionamiento comienza y termina dentro del horario laboral.

Ejemplo: de 15hs a 17hs. La tarifa será de $80

(cantidadDeHoras * precioPorHora) → (2 * 40)

-**TarifaFueraDeHorarioLaboral**:

El estacionamiento comienza y termina fuera del horario laboral.

Ejemplo: de 21hs a 23hs. La tarifa será de $0

-**TarifaInicioAntesDeHorarioLaboralYFinDentroDeHorario**:

El estacionamiento comienza antes del horario laboral pero termina dentro del horario laboral.

Ejemplo: 5am a 10am. La tarifa será de 120$

Se cobrarán solo las horas a partir de las 7am.

-**TarifaInicioDentroDeHorarioLaboralYFinFueraDeHorario**:

El estacionamiento comienza dentro del horario laboral pero termina fuera del horario laboral.

Ejemplo: 17hs a 21hs. La tarifa será de 120$

Se cobrarán solo las horas hasta las 20hs.

El SEM inicializa con un rangoHorarioLaboral que es de 7 a 20.

El SEM inicializa con un calculador que es una instancia de CalculadorDeTarifa

Cuando se calcula el costo de un estacionamiento:

-se crea una nueva instancia de RangoHorario con los datos del estacionamiento

*laHoraInicioEstacionamiento será la horaInicio del rango

*laHoraFinEstacionamiento será la horaFin del rango.

-Delego en el calculador para que me devuelva el monto, pasándole:

*RANGO HORARIO LABORAL

*RANGO HORARIO DEL ESTACIONAMIENTO

*PRECIO POR HORA

-El calculador va a verificar en que rango horario se encuentra ese estacionamiento respecto del rango horario laboral

-Determina el costo según el horario.

Patrones de diseño utilizados:

Patron Strategy

Según enunciado del TP:

"Modos manual y automático

La app puede funcionar en dos modos: manual y automático. El modo es elegido por el usuario y puede ser cambiado por él mismo en el momento en que lo desee."

Clase APP es el Contexto, según el modo de app que elija el usuario, realizara el comportamiento del modo asignado (Manual o Automático)

La interfaz ModoApp, La estrategia, declara una interfaz común a todos los algoritmos permitidos. El Contexto (App) usa esta interfaz para llamar al algoritmo definido por una EstrategiaConcreta.

Estrategias concretas:

Clase Manual, Clase Automatico. Implementan la interfaz ModoApp y cada estrategia concreta define su comportamiento/algoritmo.

Patron State

La clase App es el Contexto, según el desplazamiento del usuario, realizara alertas al usuario.

La interfaz ModoDesplazamiento es el Estado

Las Clases ModalidadConduciendo, ModalidadCaminando son estados concretos

---

Patron Strategy

La clase App es el Contexto, el usuario puede determinar si desea o no recibir notificaciones.

La interfaz ModoNotificaciones es la Estrategia,

Las clases NotifiacionesActivadas, NotifiacionesDesactivdas son Estrategias Concretas.

Patron Observer

La clase SEM es el Sujeto concreto, notificar a sus observadores cuando se realizan distintas actividades, por ejemplo, el inicio de un estacionamiento.

La clase Notificador es el Sujeto, Conoce a los observadores que serán notificados.

La interfaz Obsever, define una interfaz para actualizar los objetos que deben ser notificados ante cambios en un sujeto.