Justificaciones de diseño

Se decidió agrupar toda la información relacionada a la dirección de una **Persona** en la clase **Direccion** y para que no haya inconsistencia de datos con distintas cadenas de String, se crea la clase **Calle**. Elegimos una Clase en vez del tipo de dato **Enum** para evitar que el programa deba ser recompilado frente al cambio de nombre de alguna calle. Con respecto a las comunas, se decidió tratarlas como Integer para evitar ambigüedades.

Creamos la Clase **Usuario** con los atributos userName y password y asumimos que al estar dado de alta es al menos un usuario pasivo en la aplicación.

Creamos la Clase **Viaje** que tendrá toda la información pertinente al transeunte y al trayecto que deberá realizar. Para indicar que el viaje terminó, creamos el atributo finalizado del tipo Boolean.

Tomando como base de diseño el *Patrón Strategy*, **ReaccionIncidente** se pensó como interfaz, ya que declaran el método **reaccionar(Viaje)** pero será implementado de forma distinta según las clases que heredan de ellas, que son **AlertarCuidadores**, **LlamarPolicia**, **LlamarTranseunte** y **FalsaAlarma**.

Se implementa el *Patrón Adapter* para calcular el tiempo de demora aproximado, ya que la distancia será calculada por una API externa.

Para que el Modelo de Dominio cumpla con los requerimientos del punto 2, se modifica el atributo *demora* a *demoras* que es de tipo **TipoDemora**. Aplicamos el *Patrón Strategy* al agregar la Clase **VariasParadas**, de este modo, podrá comportarse de manera distinta si tiene una o más paradas. En cada caso, se aplica el *Patrón Adapter* ya que se conectará con la API externa para realizar los cálculos correspondientes.

Con el *Patrón Adapter* (comportamiento) ganamos:

- Mayor mantenibilidad debido al nulo acoplamiento entre el cálculo de demora del objeto **Viaje** y la API.
- Mayor cohesión de la Clase **Viaje** debido a la delegación en el adapter.
- Mayor facilidad de testeo.

Con el *Patrón Strategy* (estructural) ganamos:

- Mayor cohesión.
- Mayor mantenibilidad debido a que los comportamientos para cada incidente y para cada cálculo de demora son fácilmente localizables.
- Extensibilidad para incorporar nuevos algoritmos/formas de reaccionar ante los incidentes.