Patron adapter: Lo utilizo para conectar en mi sistema el cálculo de la distancia que va a resolver la API-REST, ya que la interfaz de la API-REST no concuerda con la de mi sistema.

Patron Strategy: Lo utilizo para poder encapsular las diferentes formas en las que se puede resolver un Incidente, y me permite también que el usuario pueda configurar lo que se va a hacer frente a un Incidente. Se requiere que se pueda realizar una misma acción, que es la de actuar frente a un incidente, pero con distintos pasos o de una forma distinta. En este caso elijo que AccionIncidente sea una interface porque cada una de las estrategias implementan su propia lógica en sus métodos.

-La configuración de que debe hacer la persona ante un incidente está dentro de la clase ConfiguracionIncidente, ya que considero que el Incidente no debe ser un atributo de la persona.

-Considero a Dirección como una clase, para evitar repetidos.

-Considero que los tipos de sexo de la persona, y del estado de los viajes, no van a cambiar, por lo que utilizo un Enum.

-Considero la clase notificación, para modelar las notificaciones que hace el sistema a las personas.

-El transeúnte y el/los cuidador/es están modelados como Persona, ya que no tienen, o tienen muy poco comportamiento en común como para crear dos clases distintas.