Decisiones:

- Con respecto a las Persona decidimos no hacer una clase abstracta Persona y que hereden una clase Transeúnte y otra Cuidador por el hecho de que una Persona puede ser en un momento transeúnte, luego cuidador y después de nuevo transeúnte por lo que se estaría borrando y creando la clase de la misma persona dependiendo su función lo que no tiene sentido. Al poder cumplir las dos funciones, no tienen distinto comportamiento por lo que se decidió realizar una sola clase Persona.
- Para la clase Viaje decidimos que esta se cree en el momento en que la persona selecciona los cuidadores y su destino final, esto implica que será el viaje el que inicie la petición de notificaciones al notificador y será el que se encargue a través de la calculadora de tiempo de especificar el tiempo de demora, también contará con un boolean para saber si se podrán enviar notificaciones al transeúnte.
- Usamos el patrón Strategy para las diferentes reacciones debido a que de esta forma se pueden agregar más formas de reacción a futuro sumando extensibilidad y mantenibilidad.
- Para las notificaciones usamos nuevamente el patrón de Strategy ya que se pueden agregar distintas formas de notificar, además permite que el Notificador queda desacoplado al sistema.
- Para el cálculo del tiempo de demora se decidió ir por un patrón Adapter ya que se nos avisa por el enunciado la necesidad de trabajar con “Distance Matrix API” de Google. Lo que seguramente implique tener que adaptarse para poder realizar dicho cálculo y obtener lo necesario para nuestro sistema.

Punto 2

- Para el punto 2, en el momento que hay que calcular las demoras aproximadas por secciones y en el caso que no se detenga se deberá hacer un cálculo aproximado total, lo pensamos de la siguiente manera. Al principio pensamos en hacer un patrón Strategy dependiendo de si detenía en las paradas o no pero luego llegamos a la conclusión de que no tenía sentido ya que en caso de no tener paradas, el tiempo de demora sería 0 y ese tipo de “estrategia” no tendría efecto. Por lo que la CalculadoraDeTiempo solo deberá corroborar si tiene paradas o no. Si tiene paradas, obtiene todos los tiempos de demora de cada una (si no se detendrá en la parada se indica 0 en el tiempoDeDemora de esa parada en específico) y se lo suma a lo que la DistanceMatrixAPI calculó para el punto inicial hasta el final. Si no tiene paradas devuelve el cálculo de la DistanceMatrixAPI.