Modelar e implementar un prototipo de sistema de parkings

Preguntas frecuentes

- **¿Puede el simulador informar un ingreso o egreso de un transitable inexistente, o sea, informar un numero de patente que no existe?**
    - No. El simulador no se equivoca ni en leer números de patente ni en identificar códigos de cochera. El simulador enviara siempre un transitable y un estacionable de la lista que le fue proporcionado al configurarlo.

- **¿A qué se refiere la letra del enunciado cuando menciona "Capacidad del parking"?**
    - La *capacidad* de un parking es la cantidad de cocheras. La *disponibilidad* es la cantidad de cocheras libres en un momento determinado.

- **¿Puedo usar los métodos de las interfaces Transitable y Estacionable para aplicar lógica de negocio?**
    - No, no deberían usarse los métodos @Override de esas interfaces (por ejemplo "esDiscapacitado()"). El simulador es una herramienta de testing que, eventualmente, será eliminado de la solución para pasar a integrarse con un sensor real del parking. La solución debería mantenerse desacoplada tanto como sea posible del simulador.

- **¿Debe considerarse que una cochera pueda tener más de 2 estados?**
    - No hace falta. Una cochera puede estar libre u ocupada únicamente.

- **¿La solución debe soportar que se agreguen nuevos Tipos de Vehículo?**
    - Si.

- **¿La solución debe soportar que se agreguen nuevos estados de tendencia del parking además de "estable", "positiva" y "negativa"?**
    - No. Aunque siempre es deseable contar con una arquitectura robusta y escalable.

- **¿Puedo apagar el log del simulador que imprime en consola?**
    - Si. La clase SimuladorTransito tiene un metodo "pausarLogger()" y otro "reanudarLogger()". Por defecto esta activo.

- **¿Puedo forzar al simulador a informar una estadía especifica, es decir ingresar un vehículo a una cochera determinada?**

- Si. El simulador permite programarle flujos de ingreso y egresos determinísticos, proporcionándole una lista predefinida de Transitables y Estacionables. Por ejemplo:

```
LinkedHashMap<Transitable, Estacionable> preasignacion = new LinkedHashMap();
listaIngresos.put(vehiculo1, cocheraA);
listaIngresos.put(vehiculo2, cocheraB);
listaIngresos.put(vehiculo3, cocheraC);
simulador.programar(new FlujoIngreso("ingreso", new Periodo(0, 1), preasignacion));
```

La misma variable `preasignacion` puede emplearse para programar un `FlujoEgreso` también.

También se puede usar la interfaz de usuario subida a aulas "SimuladorIU" para generar trafico manual.

- **¿Cómo puedo indicarle al simulador que genere anomalías?**

El simulador puede generar anomalías en los flujos de egreso únicamente. El simulador permite programar flujos de egreso con ***perfiles*** personalizados. El siguiente ejemplo programa un flujo que produce 1 anomalía cada segundo, durante 10 segundos:

```
PerfilEgreso perfilConAnomalia = PerfilEgreso.Builder()
                                    .generarAnomalia(Modo.SIEMPRE)
                                    .build();
simulador.programar(new FlujoEgreso("anomalias", new Periodo(0, 10), 10,
perfilConAnomalia));
```

El enum "MODO" acepta 3 valores: SIEMPRE, NUNCA y ALEATORIO.

Nota: para generar anomalías "HOUDINI" en el flujo de ingreso se puede utilizar un flujo preasignado como se acaba de explicar mas arriba.

- **¿Cómo puedo indicarle al simulador que genere infracciones?**

Las infracciones son generadas por flujos de ingreso. El simulador permite programar flujos de ingreso con perfiles personalizados. (Notar que los perfiles de ingreso y egreso son distintas clases). El siguiente ejemplo programa un flujo de ingreso que produce 10 infracciones en 10 segundos. Todas ellas (Modo.SIEMPRE) invadirán un estacionable de discapacitados y, adicionalmente, algunas (Modo.ALEATORIO) invadirán también un estacionable para vehículos eléctricos.

```
PerfilIngreso perfilConInfracciones = PerfilIngreso.Builder()
                                .invadirEstacionableDiscapacitado(Modo.SIEMPRE)
                                .invadirEstacionableElectrico(Modo.ALEATORIO)
                                .invadirEstacionableEmpleadoInterno(Modo.NUNCA)
                                .build();
simulador.programar(new FlujoIngreso("infracciones", new Periodo(0, 10), 10,
perfilConInfracciones));
```

- **¿Puedo incorporar una interfaz de usuario para generar mis propios flujos para testear mi solución?**
    - Aunque no está solicitado, el alumno puede incorporar cualquier desarrollo/técnica/herramienta que considere que aporte a mejorar el testing y la calidad de la entrega. También puede emplear la interfaz de usuario subida a aulas (SimuladorIU) que permite generar tráfico a manualmente. No obstante, esos agregados no serán empleados por el docente para la verificación del funcionamiento de la entrega.