Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2115 – Programación como Herramienta para la Ingeniería (II/2018)

Laboratorio 1 - Programación Orientada a Objetos

Objetivos

- Aplicar los contenidos de Programación orientada a objetos
- Realizar una simulación de eventos de la vida real mediante Programación orientada a objetos

Entrega

- **Lenguaje a utilizar:** Python 3.6
- **Lugar:** repositorio privado GitHub. Recuerde incluir todo en una carpeta de nombre **L01**
- **Fecha límite de entrega: miércoles 29 de Agosto, a las 23:59 hrs.**
- **Formato de entrega:** dos archivos con la solución propuesta, uno en formato **.py** y el otro en **.ipynb**. Deben ser exactamente iguales (el ayudante revisará cualquiera de ellos). Adicionalmente, incluya un archivo **README.md** con información útil para el ayudante corrector.

Introducción

La Línea 10 del Metro de Santiago

La ciudad de Santiago necesita con urgencia mejorar la Línea 10 (L10) de su servicio de Metro. Con este objetivo en mente, es que el gobernador de la ciudad te ha convocado a ti, un gran conocedor de este modo de transporte, para que simules el comportamiento de la L10. Este trabajo permitirá que un analista proponga, en caso de requerirse, medidas que impulsen la calidad del Metro de esta urbe.

La simulación

A continuación, se describen los elementos que debes tener en consideración para realizar la simulación pedida. Posteriormente, se entregan detalles del funcionamiento del Metro de la ciudad.

Componentes de la Línea 10 del Metro

La Línea 10 del Metro de Santiago está compuesta por 10 estaciones. Como en la mayoría de los servicios de trenes subterráneos, todas las estaciones permiten viajes en dos sentidos, por lo que cada una tiene dos andenes. Los usuarios del Metro saben claramente cuáles son las estaciones que corresponden al origen y al destino de su viaje. Además, van siempre muy atentos a las estaciones que les faltan para completar su viaje, por lo que no está contemplado que se *pasen*[1].

El Metro no podría funcionar sin la existencia de trenes. Los trenes están compuestos por una cierta cantidad de vagones y cada uno de estos posee una capacidad máxima de $n = 200$ pasajeros [pax], que debe ser respetada cuando estos aborden. Las amplias puertas de los vagones tienen un flujo máximo de $q = 25$ personas por segundo [pax/seg] (para bajar y/o subir), que también debe ser respetado en todo momento.

Funcionamiento del sistema de Metro de Santiago

Los siguientes puntos corresponden a detalles importantes que debes considerar en tu simulación.

1. El sistema de Metro dispone de 3 horarios diferentes en sus 17 horas de funcionamiento de trenes. La llegada de usuarios a cada estación se rige por una distribución *Poisson* con tasa de llegada λ en pasajeros por hora [pax/h] descrito en la tabla de a continuación.

[1] https://www.youtube.com/watch?v=VATlBrBkB0Y

Horario	**Hora de inicio**	**Hora de termino**	λ **[pax/h]**
Bajo	06:00	06:29	7000
Valle	06:30	06:59	7000
Punta	07:00	08:59	7000
Valle	09:00	17:59	7000
Punta	18:00	19:59	7000
Valle	20:00	20:44	7000
Bajo	20:45	23:00	7000

2. Las estaciones abren a las 6:00 pero las personas llegan $Poisson$ a partir de las 5:00 a una tasa de 200 [pax/h] (entre 5:00 y 5:59). Dado que las estaciones se encuentran cerradas en este horario, las personas se ponen en cola fuera de las estación. De este modo, a las 6:00 entran todas juntas y comienza el proceso de llegada descrito en la tabla.

3. Para modelar las estaciones se debe considerar que cada una tiene un nombre (mencionados más abajo) y solo dos andenes (no hay estaciones con combinación que tengan más andenes)

4. La línea posee 16 trenes en circulación. A las 6:00 se encuentran 8 de ellos en cada estación terminal. A esta misma hora comienzan a operar en intervalos de 5 minutos.

5. El tiempo de viaje entre dos estaciones consecutivas es de 4 minutos.

6. Cuando un tren llega a su estación terminal, este opera normalmente y se devuelve en la otra dirección.

7. A cada tren se le debe definir el número de vagones al inicio de la simulación y este no debe ser modificado. Para ello debe usarse la distribución $Uniforme(7, 10)$.

8. Cuando una persona llega a una estación, se le asigna su destino aleatoriamente (que no sea la estación en la que se encuentra). A partir de eso se ubica en el andén correspondiente. Las personas en punta mañana (7:00 y 8:59) se dirigen con probabilidad 0.7 a las cuatro estaciones centrales y en punta tarde (18:00 y 19:59) con 0.8 de probabilidad a las seis estaciones de los extremos (tres de cada lado) La elección de estaciones en horarios "valle" y "bajo" es equiprobable.

9. Considera que esta línea será similar a la actual Línea 6 del Metro de Santiago: los andenes tienen puertas por las que sube y baja la gente del metro. La gente hará colas en cada puerta para poder abordar el tren. Al llegar, todas las personas eligen la cola más corta. En caso de empate, elige cualquiera de las candidatas.

10. Cada vagón tiene una capacidad de 200 personas y posee exactamente una puerta por donde entra y sale la gente. El anden tiene una cantidad de puertas igual a la del tren más largo posible, es decir, diez puertas.

11. El tren siempre se ubicará al comienzo del andén. Por ejemplo, un tren de siete vagones ocupará las primeras siete puertas del andén en el sentido de circulación. El tren más largo ocupará siempre todo el andén.

12. Cuando la gente esté haciendo una fila para un vagón que está lleno, la persona se cambiará al final de cualquier fila útil con probabilidad 0.6. Una fila se dice útil si conduce a un vagón que no está lleno. Los cambios de fila toman 1 segundo y las personas que se encuentren más al final de la fila deciden primero.

13. De manera similar al punto anterior, si llega un tren corto y la persona se encuentra esperando inútilmente en una puerta a la que no llegará el tren, esta se moverá a una fila útil con probabilidad 0.7 (solo si hay filas útiles disponibles). Los cambios de fila toman 1 segundo y las personas que se encuentren más al final de las filas y más lejos del tren deciden primero que las demás. Es decir, las primeras personas en decidir serán las que estén al final de la fila de la puerta número 10.

14. La ciudad se ha vuelto muy respetuosa y en cada vagón los usuarios que suben esperan hasta que todos los pasajeros en viaje que deseen bajar puedan hacerlo (en el mismo vagón). Cuando llega el metro a un andén, bajar todos los pasajeros con destino en esa estación y luego comienzan a subir. En general, los usuarios pueden subir a menos que:

    - Todos los vagones estén repletos.
    - Se acabe el tiempo en el que el tren mantiene sus puertas abiertas.
    - Todas las personas abordaron el tren.

15. El tren abre sus puertas apenas llega a la estación y las mantendrá abiertas durante 30 segundos a partir del momento en que las abra o hasta que todos hallan podido bajar del tren. Si todos bajaron antes de 30 segundos, entonces estará 30 segundos abierta. Si se demoran más de 30 segundos, entonces cerrará sus puertas cuando baje el último. Posteriormente, el tren cerrará sus puertas y esperará hasta que la siguiente estación esté disponible (no haya un tren detenido en ella)

16. Las estaciones del metro en orden son las siguientes: “Tranca Perro”,“Nuestra señora Danielita”, “Avenida Hans Lobel”, “Plaza Mavrakis”, “Hugo Hurtado”, “San Halcón de Chicureo”, “Quinta Osornal”, “Côte d’Ivoire”, “Manquehuito”, “Hernando de Valdivia”, “Liru Sisa”

Corrección

Para la corrección de este laboratorio, se revisarán 2 ejes: modelación y funcionalidad. En el primero, se evaluará la correcta aplicación de los conceptos de herencia, agregación, composición, *properties*, etc. Para obtener el puntaje del segundo eje, deberás crear y completar una serie de funciones en un archivo llamado *function_simulator.py*, las que solicitarán que imprimas en pantalla o retornes algunos datos en específicos luego de realizar la simulación. Estas funciones son:

- `capacidad_tren(estaciones)`

  El parámetro `estaciones` es una lista de *strings*, donde cada elemento es el nombre de una estación. Cada vez que algún tren pasó por alguna de las estaciones especificadas en la lista, debes imprimir en consola la capacidad del tren antes y después de cerrar las puertas.

  El formato de los `print` es:

  ```
  Tiempo {N}: Tren {N} llega a la estación {Nombre}.
      Antes de abrir las puertas: {N} personas.
      Después de cerrar las puertas: {N} personas.
  ```

- `datos_anden(estacion, anden)`

  Esta función recibe el nombre de una estación y el número del andén. Debe retornar la cantidad de personas totales que llegaron a ese andén y el tiempo promedio de llegada de cada tren. Si el andén es igual a 1, entonces es el andén en dirección a "Tranca Perro", en otro caso es en dirección a "Liru Sisa"

- `patrones()`

  Con esta función se podrá conocer la estación donde abordaron más personas y la estación donde descendieron más individuos. Para ambos casos, deberás imprimir el nombre de la estación y la cantidad de personas que se bajaron o subieron.

- `horarios(bajo, valle, punta)`

  En el contexto de esta función, se recibirán las tasas de llegada de personas a las estaciones del Metro de Santiago. Dado esto, la función `horarios` recibe los parámetros `bajo`, `valle` y `punta`, son 3 `integers` que representan el *lambda*, en pasajeros por hora [pax/h], de la distribución Poisson que describe la llegada de personas para 3 horarios distintos. Con esta información, deberás realizar la simulación de

modo que la tasa de llegada de las personas cambie según el horario y los números entregados como parámetros, para finalmente imprimir los siguientes datos:

1. Número de personas que llegó por horario (valle, punta y bajo).
2. Estación más congestionada. Esta es aquella que tiene el menor flujo de subida.
3. Estación menos congestionada. Esta es aquella que tiene el mayor flujo de subida.

La siguiente tabla muestra un ejemplo de tasas de llegada que podría recibir la función al aplicar `horarios(500, 5000, 8000)`

Horario	**Hora de inicio**	**Hora de termino**	λ **[pax/h]**
Bajo	06:00	06:29	500
Valle	06:30	06:59	5000
Punta	07:00	08:59	8000
Valle	09:00	17:59	5000
Punta	18:00	19:59	8000
Valle	20:00	20:44	5000
Bajo	20:45	23:00	500

- `simular_verboso()`

  Debes realizar la simulación e imprimir en pantalla cada evento que ocurra. Este debe contener:

  - Tiempo de simulación (hora, minuto y segundo)
  - Descripción del evento

  Ejemplos:

  - 2:0523 - Persona 5 llega a la estación "Quinta Osornal" en dirección a Liru Sisa.
  - 4:23:32 - Persona 24 llega a su destino: Estación "Côte d'Ivoire".
  - 12:55:35 - Tren 2 sale de la estación "Hugo Hurtado" con dirección "Tranca Perro".

  Importante: esta función tendrá mayor ponderación en la nota final que las otras funciones.

A continuación se muestra algunos ejemplos de como el ayudante corregirá las funciones anteriores ejecutando un archivo `test.py` como el siguiente:

```
import function_simulator


capacidad_tren(["Tranca Perro", "Nuestra señora Danielita"]
capacidad_tren(["Liru Sisa"])
print(datos_anden("Liru Sisa", 1))
print(datos_anden("Hugo Hurtado", 0))
patrones()
simular_verboso()
horarios(1000, 1000, 1000)
horarios(500, 5000, 8000)
```

Sugerencia: programe estas funciones con los nombres aquí indicados. Si los cambia, puede tener problemas en la corrección de su laboratorio.

Política de Integridad Académica

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Ejemplos de actos deshonestos son la copia, el uso de material o equipos no permitidos en las evaluaciones, el plagio, o la falsificación de identidad, entre otros. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica en relación a copia y plagio: Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Si un alumno (grupo) copia un trabajo, se le calificará con nota 1.0 en dicha evaluación y dependiendo de la gravedad de sus acciones podrá tener un 1.0 en todo ese ítem de evaluaciones o un 1.1 en el curso. Además, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir un procedimiento sumario. Por "copia" o "plagio" se entiende incluir en el trabajo presentado como propio, partes desarrolladas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.