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

IIC2115 – Programación como Herramienta para la Ingeniería (II/2017)

Tarea 1

Objetivos

- Modelar correctamente utilizando Programación Orientada a Objetos (POO).
- Diseñar una red de metro con las estructuras de clases vistas en clases.
- Manejar *inputs* del usuario para realizar consulta sobre la red de metro
- Redactar un informe final sobre la tarea utilizando **LATEX**

Entrega

- **Lenguaje a utilizar:** Python 3.6
- **Lugar:** GitHub
- **Fecha:** x de septiembre
- **Hora:** 23:59
- **Desarrollo Individual**

Recomendación

Lean la tarea completa antes de comenzar a trabajar y fíjense en los sustantivos y verbos relacionados con el problema.

Al terminar la tarea escriban un comentario al final de su informe dando a conocer su opinión y/o su percepción de la tarea. Estos comentarios sirven para ir mejorando el curso en el transcurso del semestre por lo que se sugiere realizarlos a conciencia y no se penalizará en caso de que el comentario sea negativo.

Introducción

Nuestro Decano se aburrió de construir nuevos edificios en San Joaquín y ha comenzado un nuevo proyecto, un nuevo sistema de metro que pasa por debajo del actual y que solo es para estudiantes y funcionarios de la Universidad Católica. Para este magnate proyecto se ha comenzado a discutir cual sería una red de metro eficiente para nuestro país (parece que no le gusta la red de metro chilena) y se está debatiendo entre una red como la de Moscú, la de Hong Kong o la Boston por lo que se te ha encargado realizar un programa capaz de modelar estas lineas de metro y realizar una serie de consultas que serán vitales para decidir cual linea de metro construir.

Descripción

El objetivo de esta tarea es modelar y realizar una serie de consultas a una linea de metro. Para esto se les entregará información sobre las estaciones de metro existentes y 3 modelos distintos de como están conectadas estas estaciones.

Consultas

Su programa debe ser capaz de recibir el nombre de dos estaciones diferentes e indicar una ruta que conecte ambas estaciones. A partir de esa ruta, su programa debe permitir realizar 5 diferentes consultas. Las primeros 2 consultas serán definidas a continuacion mientras que las otras 3 serán *consultas libres*, es decir, quedan a su criterio que tipo de consulta hacer a esa ruta.

- Transbordo: Debe indicar la cantidad de transbordo existentes en esa ruta. Se define transbordo como cualquier estación que permite tomar 2 o más rutas diferentes sin considerar el cambio de sentido de viaje como una nueva ruta.
- Velocidad promedio: Debe mostrar la velocidad promedio que viaja una persona en esa ruta.

Estaciones y modelos

Junto al enunciado, se sube un código con 2 funciones:

- **obtener_informacion()**: Esta función retorna un diccionario cuyas *keys* son los nombres de las estaciones de metro y la *value* es otro diccionario con la información de esa estación. Esta información tiene el siguiente formato

```
{
    "TME": X,
    "NPA": X,
    "NPD": X,
    mas cosas
}
```

  Donde:

  - TPE: Corresponde al tiempo promedio de espera en esa estación antes de llegar a la siguiente.
  - NPA: Corresponde al número de personas que abordan el tren.
  - NPD: Corresponde al numero de personas que desalojan el tren.

  Ejemplo:

```
{
    "Baquedano": {
                    "TPE": 32,
                    "NPA": 12,
                    "NPD": 16,
                     ...
                },
    "Santa Lucia": {
                    "TPE": 67,
```

```
                    "NPA": 32,
                    "NPD": 11,
                     ...
                },
}
```

- **obtener_modelo(numero_del_modelo)**: Dado el numero del modelo, esta función retornará una lista de tuplas de la forma (estacion_1, estacion_2, x_distancia) lo cual significa que la estacion_1 está conectada a la estacion_2 a una x_distancia medida en Kiometros.

  Ejemplo:

```
[
    ("Baquedano", "Salvador", 50),
    ("Baquedano, "Parque Bustamante", 30),
    ("San Joaquín", "Pedrero", 10)
]
```

Interfaz (Consola)

Un programa no existe sin una interfaz y esta no será la excepción. Para interactuar con su programa, deberán crear un menú interactivo por consola que permita realizar todas las consultas e imprima en pantalla la información obtenida de forma ordenada. Para esto deberán solicitar *inputs* al usuario y en base a eso realizar las acciones necesarias para que el usuario pueda ver la información esperada. Puede asumir que el usuario no intentará botar su programa ingresará los datos de forma incorrecta y por lo tanto no deben preocuparse de hacer un menú que no se caiga nunca.

Informe

Para esta tarea deberá entregar un **Manual de Usuario** escrito en **Latex** donde se explique como interactuar con la interfaz de su programa, para esto deberá mostrar en el informa cada Menú existente, que *inputs* pide y las acciones a ocurrir con cada *inputs*.

```
Menú Inicial:
¿Que desea hacer, login o registrarse?
```

El menú inicial pide un string (palabra) como input el cual puede ser “login” o “registrarse”. Las acciones a realizar son:

- login: Ir al menú de ingreso
- registrarse: Ir al menú de registro

Además deberá mostrar vía ejemplos como ejecutar cada consulta implementada junto con los resultados que se generan de dicha consulta.

BONUS

Se otorgará un puntaje adicional si entre las *consultas libres* escoge algunas de las siguientes:

- Indicar otro par de estaciones de metro que posean la misma distancia que la ruta encontrada por su programa.

- Algo mas
- Algo mas
- Algo mas

Puede escoger hasta un máximo de 3 consultas.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"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.