IIC2233 Programación Avanzada (I/2015)

Tarea 1

1. Objetivos

- Aplicar los conceptos de la programación orientada a objetos para modelar un sistema.
- Repasar conceptos de búsqueda, ordenamiento y lectura de archivos de texto plano.
- Desarrollar habilidades de documentación y buenas prácticas en la escritura de código.

2. Problema

Para esta tarea deberá modelar un sistema para el manejo de una empresa de transporte. Esto involucra representar todas las entidades como objetos y relacionarlas adecuadamente. Además se deberá crear un menú general que permita el control de este sistema desde la consola de Python.
La empresa ya posee una considerable cantidad de datos en variados archivos de texto, por lo que el sistema deberá ser capaz de utilizar esta información para poblarse en un principio. Por otro lado, el usuario deberá ser capaz de realizar consultas simples al sistema y como última funcionalidad, agregar nuevos datos.

3. Especificaciones del Sistema

El sistema debe soportar el manejo de las siguientes entidades

3.1. Pasajeros

Poseen nombre, apellido y RUT.

3.2. Carga

Posee nombre, peso, volumen y tipo. El tipo puede ser delicado, peligroso o normal.

3.3. Ciudades y Terminales

La empresa trabaja con distintas estaciones a lo largo del mundo, estas poseen un nombre, se encuentran vinculadas a un tipo de transporte y soportan un tamaño específico de vehículos. Cada uno de estas terminales se encuentra en una ciudad específica.

- Aeropuerto
- Terminal de Bus
- Puerto
- Almacén de Carga

3.4. Rutas

Las rutas son las conexiones entre las ciudades. Cada una vincula a dos ciudades, posee un nombre propio, soporta un tipo específico de vehículo, tiene un largo dado y por último posee un multiplicador de costo. Esto último afecta el valor de transporte de cualquier vehículo que la utilice.

3.5. Tipos de Transporte

Cada vehículo se diferencia dentro de dos categorías. En primer lugar se hace diferencia por el tipo de objetos que transporta:

- Transporte de Carga
  A estos vehículos se les asigna un volúmen y peso máximo de carga, además de restricciones de tipo de carga (delicado, normal o peligroso).
- Transporte de Pasajeros
  A estos vehículos se les asigna una cantidad máxima de asientos.

En segundo lugar se hace diferencia por el medio en el cual trabajan:

- Aéreo
- Acuático
- Terrestre

3.6. Vehículos

La empresa posee varios tipos de vehículos. Todos poseen un nombre, tipo, tamaño, velocidad y costo de transporte. A grandes rasgos la empresa trabaja con los siguientes medios de transporte:

- Aviones - Transporte aéreo de carga y de pasajeros.
- Buses - Transporte terrestre de pasajeros.
- Camiones - Transporte terrestre de carga.
- Barcos - Transporte acuático de carga.
- Cruceros - Transporte acuático de pasajeros.

3.7. Viajes

Cada viaje establece el movimiento de un vehículo a lo largo de la red en el tiempo. Estos poseen la siguiente información:

- Código del viaje
- Terminal de partida y de llegada
- Hora de partida
- Vehículo utilizado
- Ruta utilizada

3.8. Itinerarios

Estos son los pasajes de los objetos transportados, ya sean pasajeros o carga. Establecen los viajes que tiene agendados un objeto. Simplemente asocia un identificador (RUT o nombre) del objeto con los códigos de los viajes

4. Carga de Datos

Junto con el enunciado encontrará nueve distintos archivos de texto que contienen información de las entidades antes dadas para poder poblar su sistema.

- `passengers.txt`
  Un pasajero por línea en el archivo de la forma `"nombre \t apellido \t RUT"`
- `cargo.txt`
  Un objeto por línea del archivo de la forma `" id \t nombre \t peso \t volumen \t tipo"`
- `cities.txt`
  Una ciudad por línea del archivo de la forma `"nombre \t país"`
- `hubs.txt`
  Un terminal por línea del archivo. Además se encuentran divididos por tipo de terminal a lo largo del archivo. e.g.
  `"nombre \t ciudad \t tipo \t tamaño`
  `Airports`
  `... Aeropuertos ...`
  `BusTerminals`
  `... Terminales de buses ...`
  `CargoDepots`
  `... Almacenes de carga ...`
  `Harbor`
  `... Puertos ...``"`
- `routes.txt`
  Una ruta por línea del archivo de la forma
  `"nombre \t ciudad1 \t ciudad2 \t tipo \t largo \t tamaño \t costo"`
- `vehicle_models.txt`
  Un modelo de vehículo por línea del archivo. Además se encuentran divididos por medio de transporte a lo largo del archivo. e.g.

  `Airplane`
  `modelo \t tamaño \t velocidad \t alcance \t asientos \t volumen \t peso \t tipos_carga \t costo`
  `...modelos de aviones...`

  `CargoShip`
  `modelo \t tamaño \t velocidad \t volumen \t peso \t tipos_carga \t costo`
  `...modelos de barcos...`

  `CruiseShip`
  `modelo \t tamaño \t velocidad \t asientos \t costo`
  `...modelos de cruceros...`

  `Bus`
  `modelo \t tamaño \t velocidad \t asientos \t costo`
  `...modelos de buses...`

  `Truck`
  `modelo \t tamaño \t velocidad \t volumen \t peso \t tipos_carga \t costo`
  `...modelos de camiones...`

- `fleet.txt`
  Representa la flota de vehículos que posee la empresa. Un vehículo por línea en el archivo de la forma `"modelo \t nombre"`
- `trips.txt`
  Un viaje por línea en el archivo de la forma `" id \t origen \t destino \t ruta \t hora \t vehículo"`
- `itineraries.txt`
  Un itinerario por línea del archivo. Un objeto puede tener uno o más viajes y se ven de la forma:
  `" objeto \t viajes"`
  e.g.
  ```
  object1 \t trip1
  object2 \t trip1
  object3 \t trip1 \s trip2
  object4 \t trip2
  ```

NOTA: Se recomienda mirar los archivos de texto para aclarar dudas.

5. Funcionalidades del Sistema

Una vez poblado el sistema, el usuario deberá poder hacer algunas consultas para **obtener datos**; **crear** nuevos vehículos, pasajeros, carga y viajes; y por último **agendar** nuevos viajes a itinerarios y **cancelar** los pasajes agendados a un objeto. Todos los datos creados o modificados deben permanecer en el sistema (i.e: al iniciar el sistema se conoce la misma información que cuando lo cerramos la última vez). Todas estas funcionalidades deben respetar las restricciones establecidas previamente y la integridad del sistema.
En el caso de que algún nuevo vehículo o dato no cumpla las restricciones, no se llevará a cabo la acción y se deberá imprimir en consola la razón. *e.g. 'No se puede agregar carga peligrosa al vehículo ABC123!', 'El vehículo ABC123 se encuentra lleno y no puede cargar más pasajeros!'*

NOTA: Esta interacción será por consola con un menú a su elección. Es importante que sea simple, clara y que valide la entrada por consola sin caerse. Recuerde que el usuario podría ingresar datos no válidos. Puede asumir que los archivos que leerá existen y tienen el formato correcto.

5.1. Consultas

- Dado un identificador único (RUT, nombre, etc. lo que usted estime más conveniente), imprimir los atributos de un vehículo, ruta, pasajero, carga o viaje.
- Dada una fecha y hora, imprimir los datos de un objeto específico en ese instante
  - Persona o Carga: Lugar en el que se encuentra
  - Vehículo: Lugar en el que se encuentra y el contenido
- Dadas dos ciudades c_1 y c_2 responder:
  - Es posible o no llegar de c_1 a c_2 (la consulta debe entregar respuesta en menos de 2 minutos[1]).
  - Para un objeto (pasajero o carga) y una fecha de inicio dada, mostrar las opciones **factibles** de viaje entre c_1 y c_2 (la consulta debe entregar las siguientes 3 respuestas en menos de 5 minutos[2]):

[1] Sujeto a cambios avisados oportunamente
[2] Sujeto a cambios avisados oportunamente

- serie de viajes más barata
- serie de viajes más corta (en distancia)
- serie de viajes más rápida

5.2. Cálculos

- Dada una fecha y hora, calcular la distancia que ha recorrido un vehículo, pasajero o carga específicos.
- Para un pasajero o carga específicos imprimir el itinerario de viaje. Esto implica imprimir todos sus "pasajes": origen, destino, horario de salida, horario de llegada, nombre del viaje, vehículo, costo.

  Cómo calcular el costo de un pasaje:
  El costo base de un viaje será:

  $$costo\_base = largo\_ruta \cdot costo\_ruta \cdot costo\_vehículo$$

  Luego:

  - Para Pasajeros:
    $costo = costo\_base$
  - Para Carga:
    $costo = costo\_base \cdot max\{\frac{peso}{100}, volumen\}$

6. Restricciones y alcances

- Tu programa debe ser desarrollado en Python 3.4
- Su código debe seguir la guía de estilos `PEP8`
- Si no se encuentra especificado en el enunciado, asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[3] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- La revisión de la tarea será realizada con distintos archivos `.txt`.
- Debe adjuntar un archivo `README.txt` donde comente sus alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*.
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común.
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por encima de ningún otro.

7. Puntaje

- (3.00 pts) Modelación OOP (clases, herencia, polimorfismo, properties, uso de objetos)
- (1.00 pts) Lógica del sistema (archivos, consola, validación de datos)
- (2.00 pts) Consultas y Algoritmos

Días antes de la entrega se subirá el detalle de cada item.

[3]Hasta −5 décimas.

8. Entrega

- **Fecha/hora:** Jueves 2 de Abril de 2015 / 23:59.
- **Lugar:** GIT - Carpeta: Tareas/Tarea_01

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).