PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 - PROGRAMACIÓN AVANZADA

Actividad 02

1^o semestre 2018

22 de marzo

Estructuras de datos: *Built-ins*

Introducción

¡Oh no! La aerolínea de PrograJets PAW (*Programación Avanzada Wings*) ha tenido muchos problemas logísticos debido a un ataque que borró gran parte del código que gestiona su sistema de reservas y asignación de asientos. En un intento por resolver el problema de forma definitiva, te han contratado para completar el código en Python, con la finalidad que funcione **de forma eficiente**.

Debido a que el sistema estuvo caído mucho tiempo, las reservas que no pudieron ser procesadas fueron almacenadas en un archivo. Tu trabajo consiste en reestablecer el sistema. Para lograr esto, tienes que procesar las reservas acumuladas **según su orden de llegada**.

Entidades

En el proceso de reserva y asignación de asientos existen tres actores principales que deben gestionarse en estructuras de datos adecuadas:

- **Clientes:** tienen un nombre y un número de pasaporte único.
- **Vuelos:** tienen un identificador único, un origen, un destino y un registro de asientos. Todos los vuelos tienen capacidad para máximo **cinco** pasajeros, puesto que son PrograJets de escaso tamaño.
- **Reservas:** corresponden a solicitudes de asientos por parte de los clientes. Cada reserva se define con el número de pasaporte del cliente que la realizó, el identificador del vuelo y el número del asiento que el cliente escogió. Un mismo cliente puede solicitar asientos en varios vuelos distintos, pero sólo un asiento por vuelo. Las reservas pueden ser aceptadas o rechazadas. Mientras el sistema estuvo caído se acumularon muchas de estas reservas.

Proceso de asignación de asientos

Su sistema tiene que registrar las reservas no procesadas; es decir, aquellas que quedaron en el archivo de solicitudes acumuladas y que aún no han sido asignadas/rechazadas. Deben eliminarse las reservas de este registro a medida que se van procesando. Este proceso consiste en la asignación de asientos en los vuelos según orden de llegada de la reserva. El algoritmo es simple: si se solicita un asiento disponible, este debe ser asignado al cliente solicitante. En caso de que el asiento ya esté ocupado, se debe rechazar la reserva.

Clase PAWControl

El código de la clase `PAWControl` fue lo único que sobrevivió al ataque. Sin embargo, sus métodos para la lectura de datos de los usuarios, reservas y vuelos están incompletos. También se borraron métodos de **consultas** de estado del sistema de asientos. Se espera que implementes y completes los métodos de esta clase, pudiendo agregar otros adicionales si lo estimas conveniente.

Archivos

Tu trabajo se construirá sobre cuatro archivos.

- `main.py`: es el archivo que debes modificar para completar las funcionalidades pedidas.
- `clients.txt`: cada línea del archivo es de la forma `client_name,passport_number`.
- `bookings.txt`: cada línea del archivo es de la forma `passport_number,flight_id,seat_number` y las líneas están ordenadas según orden de llegada, donde la primera línea es la primera reserva recibida.
- `flights.txt`: cada línea del archivo es de la forma `flight_id,origin,destination`.

Consultas

Una vez que el sistema es recuperado y se asignan los asientos, tu programa debe ser capaz de responder las consultas indicadas al final del archivo `main.py` (**sin ser modificadas**). Para esto, debes completar los siguientes métodos de la clase `PAWControl`:

- `passengers_to_destination(destination)`: recibe el nombre de un destino y retorna una lista con los números de pasaportes de todos los pasajeros que viajan a dicho destino **sin que estos se repitan**.
- `rejected_bookings()`: retorna una lista de tuplas de la forma `(passport_number,client_name,flight_id,seat_number)` para cada reserva rechazada.

Afortunadamente no se ha perdido todo, y el sistema aún cuenta con la siguiente consulta que puede ayudarte a escoger tus estructuras:

- `passengers_list(flight_id)`: recibe el id de un vuelo existente y retorna una lista con todos sus pasajeros asignados. Cada elemento de la lista es una tupla de la forma: `(seat_number,passport_number,client_name)`.
  Esta lista está en orden creciente según el número del asiento. En los casos donde no hay pasajero asignado a un asiento, se entrega `(seat_number,----,----)` para ese asiento.

Una vez que completes los métodos necesarios, se espera que el output del archivo `main.py` entregado sea el siguiente (el listado de pasaportes de la segunda consulta puede estar en otro orden):

```
----- Passenger list -----
------ Flight 009 ------
('1', '10620242-2', 'Elsa Capunta')
('2', '----', '----')
('3', '----', '----')
('4', '22315433-5', 'Armando Torres')
('5', '24502023-2', 'Elba Zurita')

----- Destination -----
------ Concepción ------
18822616-1
```

```
18321654-7
10620242-2
25831580-k
25732416-0
25672177-6
11249072-9

----- Rejected Bookings -----
('24502023-2', 'Elba Zurita', '002', '4')
('18123456-0', 'Lola Mento', '006', '4')
('24949773-k', 'Armando Mocha', '004', '5')
('26912362-6', 'Zacarias Flores del Campo', '004', '3')
('18409196-9', 'Aquiles Brinco', '008', '1')
('17332754-1', 'Susana Orio', '001', '5')
('26630809-0', 'Elba Calao', '003', '3')
('11249072-9', 'Fernando Fernandez', '001', '4')
```

Notas

- En esta actividad es importante que utilicen la estructura de datos más adecuada para cada parte del problema que requiera el uso de una. Por ejemplo, si utilizan listas para todas las estructuras, no recibirán puntaje completo.
- No está permitido modificar el código de *testing* al final del archivo `main.py`. Los métodos de consulta que debes implementar y el método `__init__` de `PAWControl` deben cumplir con las firmas (nombre del método y argumentos) y valores a retornar señalados.

Requerimientos

- (3,0 pts.) Almacenamiento de info desde archivos usando estructuras adecuadas
  - (1,0 pt.) Estructura para almacenar clientes y su información
  - (1,0 pt.) Estructura para almacenar vuelos y su información
  - (1,0 pt.) Estructura para almacenar reservas sin procesar y su información
- (1,4 pts.) Procesamiento de las reservas
  - (0,4 pts.) Asignación de asientos por orden de llegada de la reserva
  - (0,6 pts.) Actualización de asientos asignados por vuelo
  - (0,4 pts.) Almacenamiento de reservas rechazadas
- (1,6 pts.) Implementación de consultas faltantes usando estructuras adecuadas
  - (0,8 pts.) `passengers_to_destination(destination)`
  - (0,8 pts.) `rejected_bookings()`

Entrega

- **Lugar:** En su repositorio privado de GitHub entregado por el curso, en la **carpeta** `Actividades/AC02/`
- **Hora:** 16:30 horas