Dr. S. Alper SERT
alper.sert@ceng.metu.edu.tr
Cihad TEKİNBAŞ
ctekinbas@ceng.metu.edu.tr

Assignment 03

METU CENG310 Fall 2023-2024
Data Structures and Algorithms with Python

Start Date: December 22th, 2023
Due Date: December 29th, 2023

1 Introduction

In this assignment, you are expected to implement flight (`Flight`) and flight database classes (`FlightDatabase`) which is described as follows.

The (`Flight`) class encapsulates individual flight details, including origin, destination, date, time, flight number, seat availability, duration, and fare. It provides methods for checking and updating seat availability, booking and canceling seats, and calculating flight duration.

The (`FlightDatabase`) class serves as a structured container for managing a collection of flights. Utilizing a SortedTableMap, it facilitates efficient storage and retrieval of flights. The class includes methods for adding flights, finding flights within time ranges, displaying all flights, and reading flights from a file. Additionally, it extends functionality by offering seat management and flight duration calculation methods. Overall, the FlightDatabase class provides a comprehensive interface for interacting with and managing flight data effectively.

The SortedTableMap class is a sorted map implementation, maintaining entries in sorted order based on keys. It extends the MutableMapping interface and utilizes a list of key-value pairs sorted for efficient search and retrieval operations. The class includes standard mapping methods and additional functionalities such as finding minimum and maximum keys, key-based searches, and key range queries. This makes it well-suited for scenarios requiring ordered key retrieval and efficient key-based operations.

1.1 The `Flight` Class

The `Flight` class represents an individual flight within a flight database.

`Flight` class should have a constructor accepting attributes of flight:

- `origin`: The origin airport code.
- `destination`: The destination airport code.
- `date`: The departure date in the format ”ddMon”(e.g., ”05May”).
- `time`: The departure time in the format ”hh:mm”(e.g., ”09:30”).
- `flight_number`: The unique identifier for the flight.
- `seats_first`: The number of available seats in the first class.
- `seats_coach`: The number of available seats in the coach class.
- `duration`: The duration of the flight in the format ”XhYm”(e.g., ”2h30m”).
- `fare`: The fare for the flight.

`Flight` class should have the following accessor functions.

`__init__(...)`

`__init__(...)` Initializes a new `Flight` instance with the specified attributes.

`__lt__(other)`

`__lt__(other)` Implements the less-than comparison for sorting based on origin, destination, date, and time.

`check_seat_availability(class_type)`

`check_seat_availability(class_type)` Returns the number of available seats for the specified class type (`’first’ or ’coach’`).

`book_seat(class_type)`

`book_seat(class_type)` Books a seat for the specified class type and updates the available seats.

`cancel_booking(class_type)`

`cancel_booking(class_type)` Cancels a booking for the specified class type and updates the available seats.

`calculate_flight_duration()`

`calculate_flight_duration()` Calculates and returns the total duration of the flight as a `timedelta` object.

1.2 The `FlightDatabase` Class

The FlightDatabase class encapsulates a set of methods for efficient management and interaction with a collection of flight.

`FlightDatabase` class should have the following accessor functions.

`__init__(self)`

`__init__(self)` Initializes an instance of the`FlightDatabase` class, creating an internal`SortedTableMap` to store flights.

`add_flight(self, flight)`

`add_flight(self, flight)` Adds a Flight object to the database using the flight's key`(origin, destination, date, time)`.

`find_flights(self, origin, destination, date, time_start, time_end)`

`find_flights(self, origin, destination, date, time_start, time_end)` Finds flights within a specified time range based on origin, destination, date, and time.

`display_all_flights(self)`

`display_all_flights(self)` Displays all flights in the database.

`read_flights_from_file(self, filename)`

`read_flights_from_file(self, filename)` Reads flight data from a CSV file and adds it to the database.

`check_seat_availability(self, origin, destination, date, time, class_type)`

`check_seat_availability(self, origin, destination, date, time, class_type)` Checks seat availability for a specific flight and class type.

`book_seat(self, origin, destination, date, time, class_type)`

`add_flight(self, flight)` Books a seat for a specific flight and class type, updating available seats..

`cancel_booking(self, origin, destination, date, time, class_type)`

`cancel_booking(self, origin, destination, date, time, class_type)` Cancels a booking for a specific flight and class type, updating available seats.

`calculate_flight_duration(self, origin, destination, date, time)`

`add_flight(self, flight)` Calculates the flight duration for a specific flight.

Example: An example scenario involving adding a flight, finding flights within a time range, checking seat availability, displaying all flights, booking a seat, and calculating flight duration within a 'FlightDatabase' instance.

1. **Input:**
   - Adding a Flight:

     ```
             flight_db.add_flight(Flight("JFK", "LAX", "10Dec", "14:30", "DL123", 10, 50, "2h30m", 300))

     ```
   - Finding Flights:

     ```
             flights = flight_db.find_flights("JFK", "LAX", "10Dec", "12:00", "18:00")

     ```
   - Checking Seat Availability:

```python
        seats_available = flight_db.check_seat_availability("
    JFK", "LAX", "10Dec", "14:30", "first")

```

2. **Output:**

   - Displaying All Flights:

```python
        flight_db.display_all_flights()

```

   - Booking a Seat:

```python
        booking_status = flight_db.book_seat("JFK", "LAX", "10
    Dec", "14:30", "first")

```

   - Calculating Flight Duration:

```python
        duration = flight_db.calculate_flight_duration("JFK",
    "LAX", "10Dec", "14:30")

```

An example input file content:

```
JFK,LAX,10Dec,14:30,DL123,10,50,2h30m,300
SFO,ORD,12Dec,09:45,UA789,5,30,3h15m,250
LAX,SFO,15Dec,18:00,AA456,15,60,1h45m,200
ORD,JFK,20Dec,12:15,DL456,8,40,2h00m,280
```

2 Delivery Instructions

Please hand in your module as a single file named as `flight.py` over ODTUClass by 11:59pm on due date. An Assignment-03 page will be generated soon after the start date of this assignment. Should you have any questions pertaining to this assignment, please ask them in advance (rather than on the due date) for your own convenience. Whatever IDE you use, you have to make sure that your module could be run on a Python interpreter:

```python
if __name__ == "__main__":
    # Example usage
    flight_db = FlightDatabase()

    # Read flights from a file
    flight_db.read_flights_from_file("flights.csv")

    # Display all flights in the database
    flight_db.display_all_flights()

    # Example usage of additional methods
    print("\nChecking seat availability for UA789 on 09May at 14:45 (
  First Class):")
    print(f"Available seats: {flight_db.check_seat_availability('UA789
  ', 'SFO', '09May', '14:45', 'first')}")

    print("\nBooking a seat for UA789 on 09May at 14:45 (Coach Class):
  ")
    if flight_db.book_seat('UA789', 'SFO', '09May', '14:45', 'coach'):
        print("Booking successful.")
    else:
        print("Booking failed. No available seats.")

    print("\nCancelling a booking for UA789 on 09May at 14:45 (First
  Class):")
```

```python
    if flight_db.cancel_booking('UA789', 'SFO', '09May', '14:45', 'first'):
        print("Cancellation successful.")
    else:
        print("Cancellation failed. No booking found.")

    print("\nCalculating flight duration for UA789 on 09May at 14:45:")
    duration = flight_db.calculate_flight_duration('UA789', 'SFO', '09May', '14:45')
    print(f"Flight duration: {duration}")
```