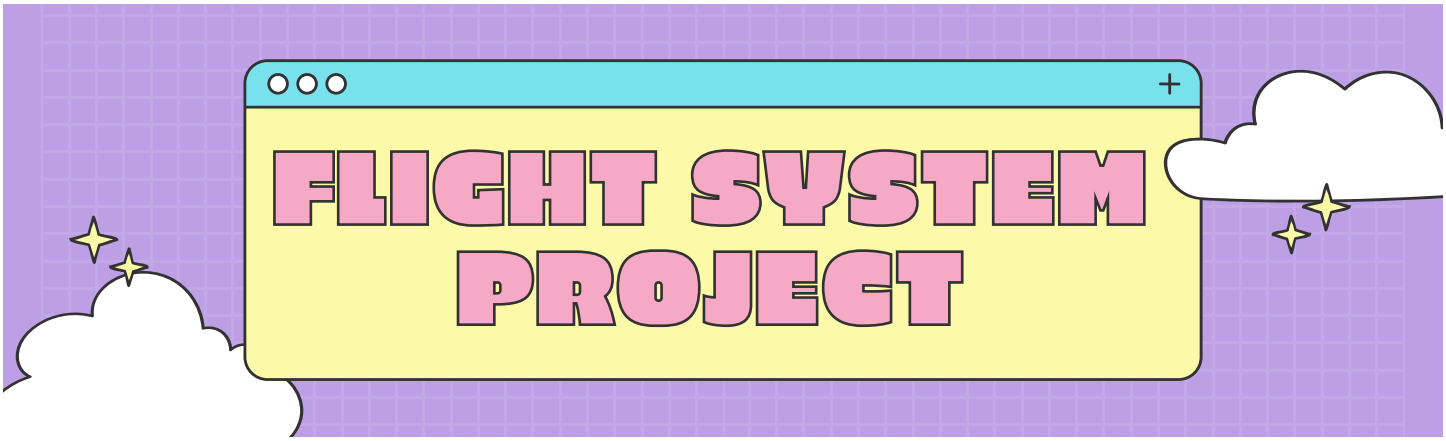


Introduction:

The flight scheduling system is designed to manage flights between cities, allowing for the addition, removal, and querying of flights. In this lab project, we have chosen to implement the flight scheduling system using an AVL tree as the underlying data structure. The AVL tree is a self-balancing binary search tree that ensures the height of the tree remains balanced, providing efficient insertion, deletion, and search operations. By using an AVL tree, we can maintain the flights in a sorted order, allowing for quick retrieval and efficient searching.

An explanation of the algorithm behind each operation:

The FlightScheduler class provides a flight scheduling system with various operations. Here's an explanation of the algorithm behind each operation:

1. Constructor: The no argument constructor initializes the FlightScheduler object. The file argument constructor allows the FlightScheduler class to be initialized with flight data read from a file. Each line in the file represents a flight, and the constructor creates Flight objects and inserts them into the AVL tree for further flight scheduling operations.

2. addFlight method:

- Create a new Flight object with the provided details.
- Check if the flight already exists in the AVL tree using the isInTree method.
- If the flight already exists, throw a FlightAlreadyExistsException.
- Otherwise, insert the flight into the AVL tree using the insertAVL method.

3. removeFlight method:

- Iterate over the flights in the AVL tree.
- Find the flight with the given flight number.
- If the flight is found, delete it from the AVL tree using the deleteAVL method.
- If the flight is not found, throw a FlightNotFoundException.

4. searchFlights method:
   - Create an empty singly linked list (SLL) to store matching flights.
   - Iterate over the flights in the AVL tree.
   - Check if the flight's origin and destination match the provided values.
   - If a flight matches, add it to the SLL.
   - Convert the SLL to an array and return the array of matching flights.

5. getFlightDetails method:
   - Iterate over the flights in the AVL tree.
   - Find the flight with the given flight number.
   - If the flight is found, return its details.
   - If the flight is not found, throw a FlightNotFoundException.

6. getFlightsSummary method:
   - Check if the AVL tree is empty using the isEmpty method.
   - If it is empty, throw a NoFlightsFoundException.
   - Print a table header for the flight summary.
   - Iterate over the flights in the AVL tree.
   - Retrieve the details of each flight and print them in a formatted manner.

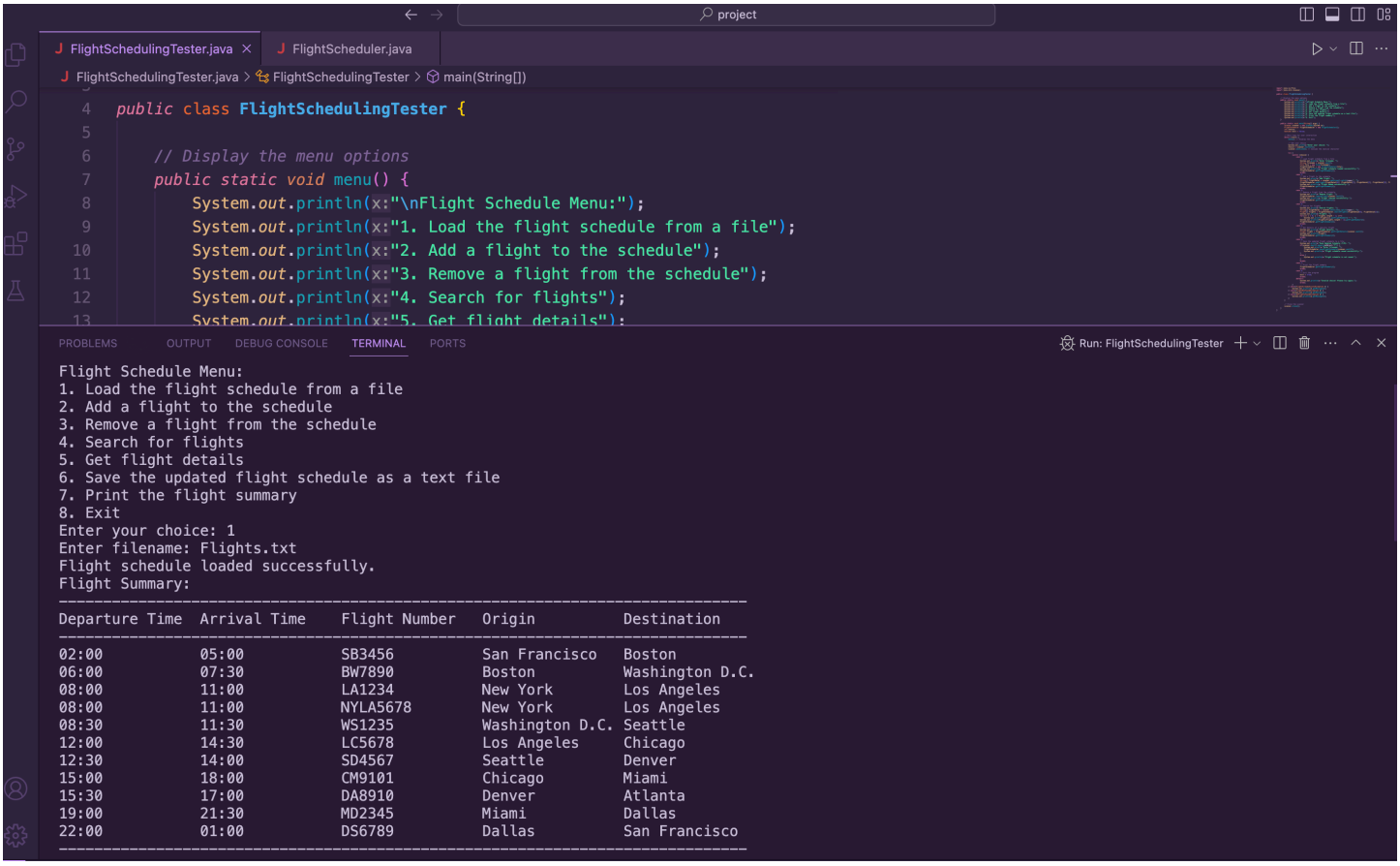
7. saveFlightSchedule method:
   - Open a PrintWriter object to write data to the specified file.

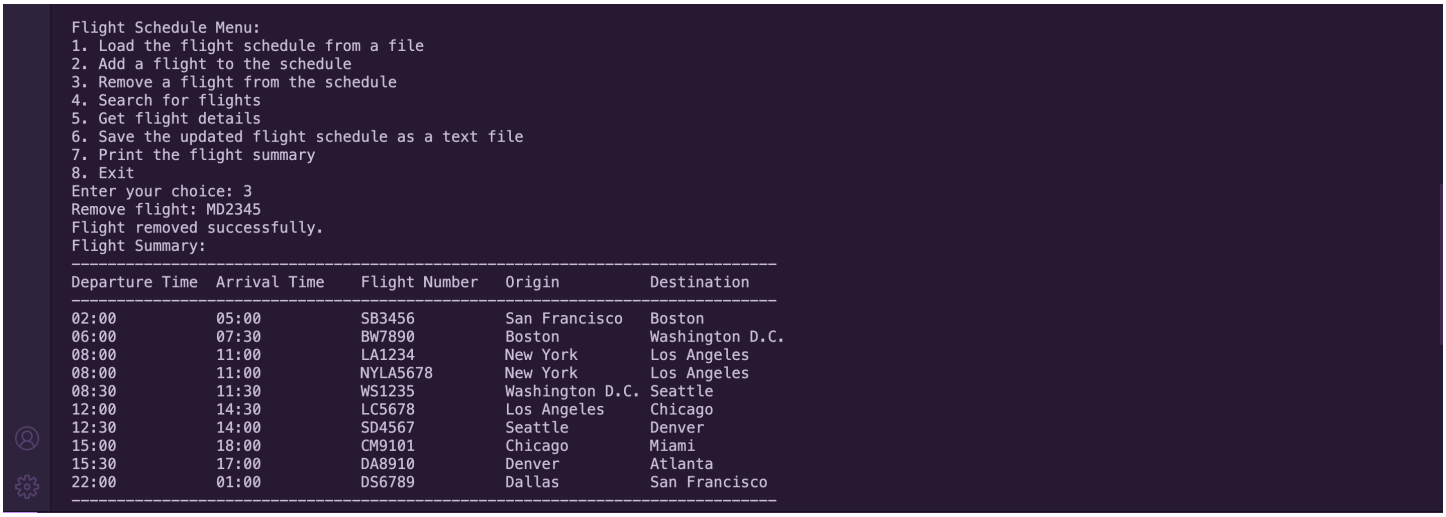
- Iterate over the flights in the AVL tree.

- Write the details of each flight to the file in a comma-separated format.

Note: The Flight class provides a representation of a flight with its attributes and necessary methods. The AVLTree class is used to store and manage the flights in a self-balancing binary search tree (AVL tree) data structure. The SLL class represents a singly linked list used in the searchFlights method.

Test Cases:

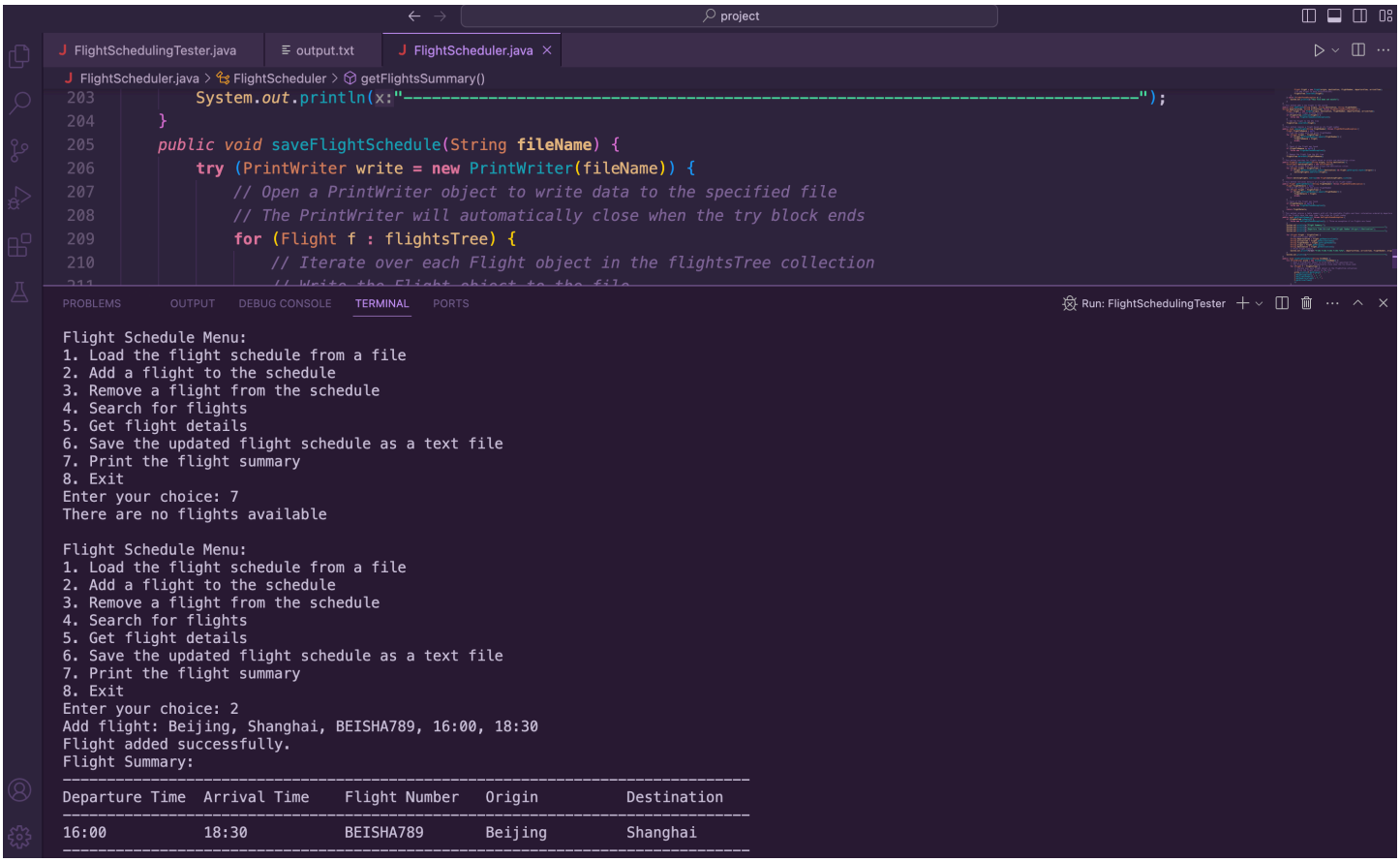




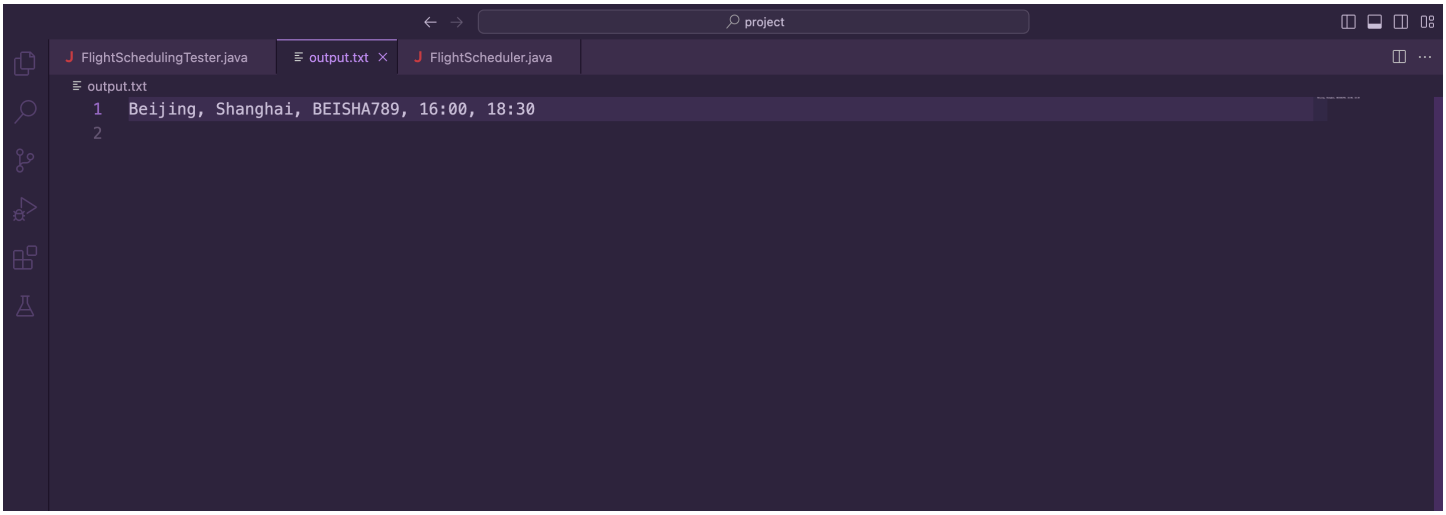
```
Flight Schedule Menu:
1. Load the flight schedule from a file
2. Add a flight to the schedule
3. Remove a flight from the schedule
4. Search for flights
5. Get flight details
6. Save the updated flight schedule as a text file
7. Print the flight summary
8. Exit
Enter your choice: 4
Search flights: New York, Los Angeles
Flights: LA1234, NYLA5678
Flight Summary:
--------------------------------------------------------------------------------
Departure Time  Arrival Time    Flight Number   Origin          Destination
--------------------------------------------------------------------------------
02:00           05:00           SB3456          San Francisco   Boston
06:00           07:30           BW7890          Boston          Washington D.C.
08:00           11:00           LA1234          New York        Los Angeles
08:00           11:00           NYLA5678        New York        Los Angeles
08:30           11:30           WS1235          Washington D.C. Seattle
12:00           14:30           LC5678          Los Angeles     Chicago
12:30           14:00           SD4567          Seattle         Denver
15:00           18:00           CM9101          Chicago         Miami
15:30           17:00           DA8910          Denver          Atlanta
22:00           01:00           DS6789          Dallas          San Francisco
--------------------------------------------------------------------------------
```

```
Flight Schedule Menu:
1. Load the flight schedule from a file
2. Add a flight to the schedule
3. Remove a flight from the schedule
4. Search for flights
5. Get flight details
6. Save the updated flight schedule as a text file
7. Print the flight summary
8. Exit
Enter your choice: 5
Flight details: LA3434
The flight is not found
```

```
Flight Schedule Menu:
1. Load the flight schedule from a file
2. Add a flight to the schedule
3. Remove a flight from the schedule
4. Search for flights
5. Get flight details
6. Save the updated flight schedule as a text file
7. Print the flight summary
8. Exit
Enter your choice: 2
Add flight: Paris, Rome, PARROM789, 18:00, 20:30
Flight added successfully.
Flight Summary:
--------------------------------------------------------------------------------
Departure Time  Arrival Time    Flight Number   Origin          Destination
--------------------------------------------------------------------------------
02:00           05:00           SB3456          San Francisco   Boston
06:00           07:30           BW7890          Boston          Washington D.C.
08:00           11:00           LA1234          New York        Los Angeles
08:00           11:00           NYLA5678        New York        Los Angeles
08:30           11:30           WS1235          Washington D.C. Seattle
12:00           14:30           LC5678          Los Angeles     Chicago
12:30           14:00           SD4567          Seattle         Denver
15:00           18:00           CM9101          Chicago         Miami
15:30           17:00           DA8910          Denver          Atlanta
18:00           20:30           PARROM789       Paris           Rome
22:00           01:00           DS6789          Dallas          San Francisco
--------------------------------------------------------------------------------
```



The updated flight schedule is saved:



Explanation of the challenges faced while doing this project and how I overcame them:

Choosing an Efficient Data Structure:

One challenge I faced was selecting an efficient data structure for managing flight data. To overcome this challenge, I researched different data structures and their characteristics, such as heaps, linked lists, hash tables, and binary search trees. I considered factors like search and insertion time complexity, memory usage, and the ability to maintain data integrity. By evaluating these factors and understanding the requirements of my project, I was able to choose the appropriate data structure, such as an AVL tree, to efficiently store and retrieve flight information.

Time Management:

Another challenge I encountered was managing my time effectively, especially when I had other projects to work on. To overcome this challenge, I prioritized my tasks and allocated specific time slots for each project. Breaking down the project into smaller, manageable tasks and creating a schedule or timeline helped me stay organized and focused. Additionally, I practiced good time management techniques, such as setting deadlines, setting aside dedicated time for each project, and avoiding multitasking, to ensure I made progress on my flight scheduling project while balancing other commitments.

Solving Problems:

During the project, I encountered various problems or bugs that needed to be resolved. To overcome these challenges, I followed a systematic problem-solving approach. This involved identifying the specific issue, gathering relevant information or data to understand the problem better, and analyzing the code or logic to pinpoint the root cause. I used debugging tools, printed debug statements, or sought help from online resources or colleagues to find solutions. By applying critical thinking, troubleshooting skills, and leveraging available resources, I was able to address the problems and make necessary adjustments to my code or implementation.

Overall, by carefully considering data structure choices, effectively managing my time, and employing problem-solving strategies, I was able to navigate the challenges and successfully progress in my flight scheduling project.