OBJECT ORIENTED PROGRAMMING
FINAL PROJECT REPORT

Explanation to the Header file:

The codeprovided in the zip file represents an Airline Flight System implemented The following points describes the structure and functionality of the header file and the code.

1. Libraries that are included:

The Libraries are included in the code are the standard C++ libraries such as `iostream`, `string`, `stdio`, `ctime`, and `sstream`. It also includes a non-standard library `conio.h`, which is commonly used for console input/output operations.

2. Helper Functions:

The given file has some of the helper functions that helps to display the menu and the different other things on the console.

- **`displayMainMenu()`:** Displays the main menu of the airline system.
- **`displayUserMenu()`:** Displays the user menu for flight operations.
- **`display_Admin_Menu()`:** Displays the admin menu for administrative operations.
- **`verifyPassword()`:** Checks if a password meets certain criteria (length >= 8, contains uppercase, lowercase, digit, and special characters).

3. Classes:

a. `Passport`:

- Represents a passport with properties like `type` (Local or Foreign) and `hasVisa` (boolean).
- Provides methods to check if the passport has a valid visa and to stamp the visa.

b. `User`:

- Base class for both `Admin` and `Passenger`.
- Contains common properties like `username`, `password`, `name`, and `cnic` (CNIC: Computerized National Identity Card).
- Provides getters and setters for these properties.

c. `Admin`:

- Derived from `User` class.
- Represents an admin user of the airline system.

d. `Passenger`:

- Derived from `User` class.

- Represents a passenger user of the airline system.
- Contains properties like `accountNumber`, `accountType`, and `Account_Balance`.
- Provides methods to set the account number and account type.
- Inherits the `displayUserMenu()` method from the base class and adds some additional menu options specific to passengers.

e. `Finance_Department`:

- Represents the finance department of the airline.
- Contains a property `name` (not used in the provided code).
- Provides a method `verify_Payment()` to verify passenger payments based on their account number.

f. `Flight`:

- Represents the attributes of the flight such as
  - source
  - destination
  - availableSeats
  - departureTime
  - arrivalTime
  - Duration
  - passengers
  - total_passenger
- Provides methods to get and set these properties, print flight details, book a seat for a passenger, and remove a passenger.
- The `bookSeat()` method is overloaded to support booking a seat with or without specifying a passenger object.

g. `Plane`:

- Represents the properties of the plane such as
  - id
  - isInternational
  - flights
  - numFlights
- Provides methods to add a flight, print the flight schedule, and book a seat on a flight.

Explanation:

- The code starts by including necessary header files and defining some helper functions.
- It then defines the classes with their member variables and member functions.

- The main function handle the flow of the program, including menus, user input, and interaction with the classes and their objects.
- The code focuses on the functionality related to user registration, login, flight scheduling, booking, cancellation, and viewing details.
- It also includes basic validations, such as password verification and checking for valid visa status for international flights.
- The code utilizes concepts of inheritance (Admin and Passenger classes derived from the User class), composition (Plane class having an array of Flight objects), and pointers (Passport object in the Passenger class).

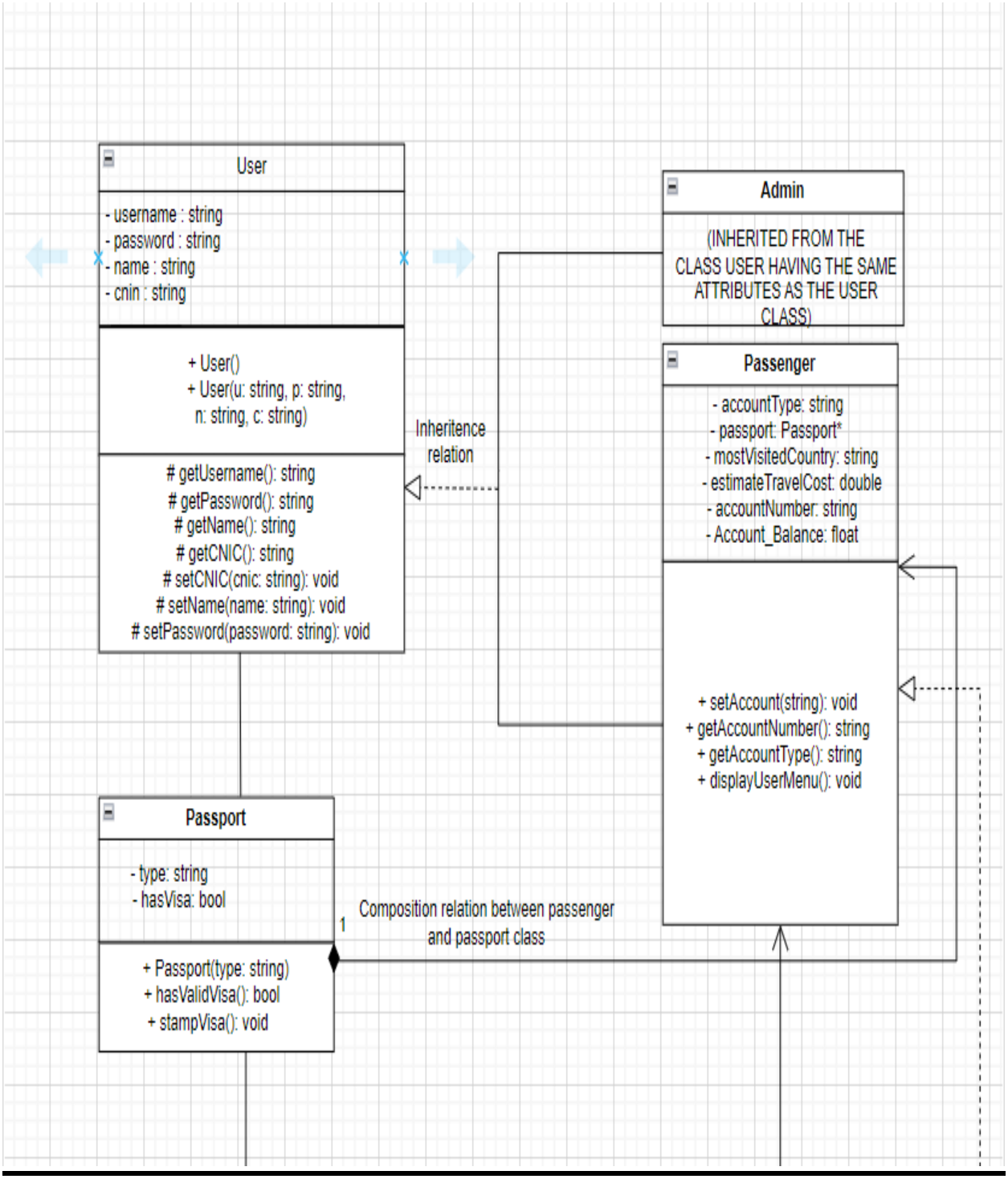
Aggregation, Composition and Inheritance:

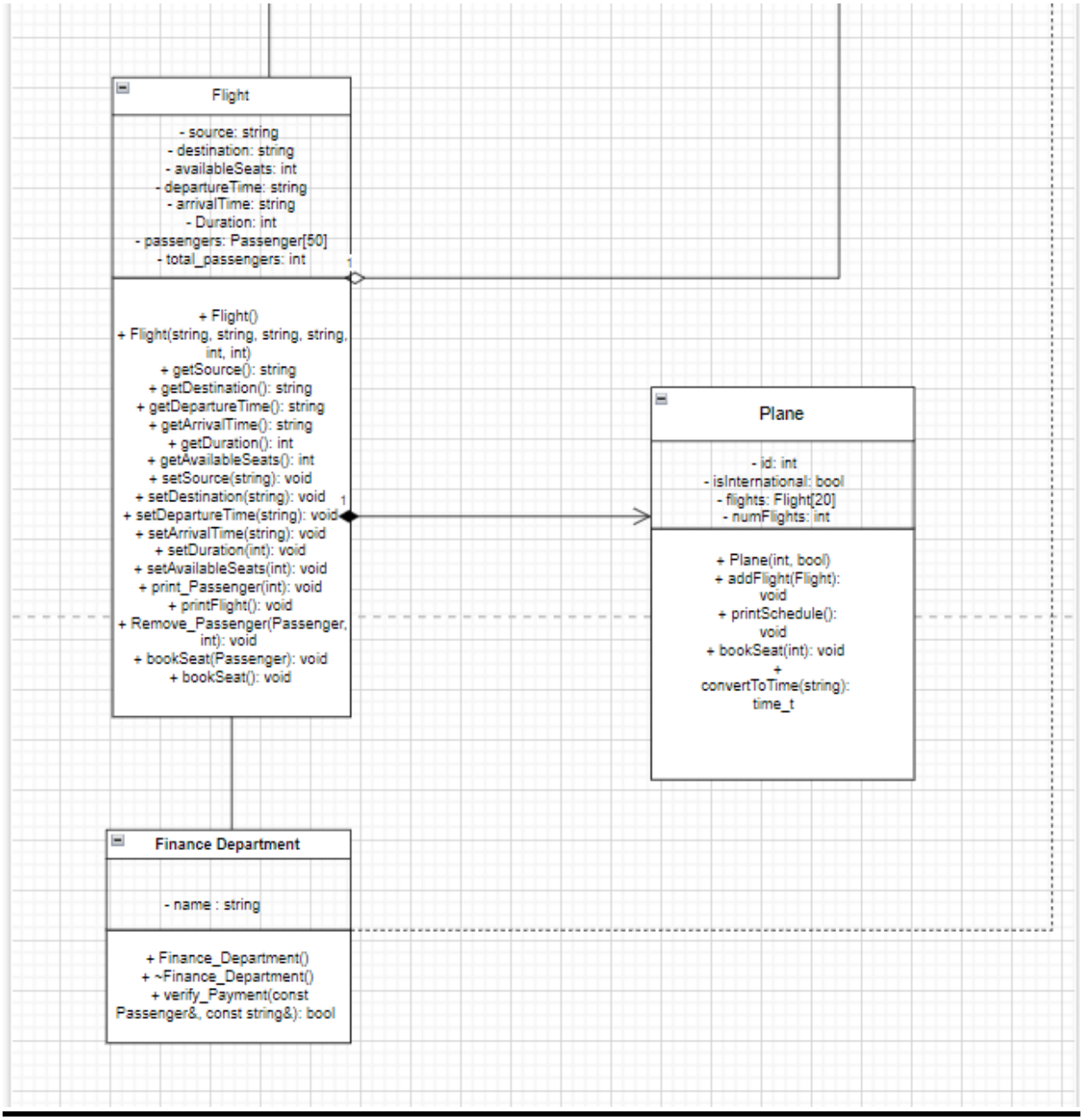
The inheritance relationship is represented by the arrow pointing from the subclass to the superclass. Aggregation is represented by a line with a diamond-shaped arrowhead pointing towards the container class for example Flight class contains an array of Passenger objects) and this shows an aggregation relation between these two classes. Composition is represented by a line with a filled diamond-shaped arrowhead pointing towards the container class for example plane contains an array of the flight objects and these both classes are showing the composition relationship.

Link to the class Diagram:

https://app.diagrams.net/#LUntitled%20Diagram

Class Diagram




Flight
- source: string
- destination: string
- availableSeats: int
- departureTime: string
- arrivalTime: string
- Duration: int
- passengers: Passenger[50]
- total_passengers: int
+ Flight()
+ Flight(string, string, string, string, int, int)
+ getSource(): string
+ getDestination(): string
+ getDepartureTime(): string
+ getArrivalTime(): string
+ getDuration(): int
+ getAvailableSeats(): int
+ setSource(string): void
+ setDestination(string): void
+ setDepartureTime(string): void
+ setArrivalTime(string): void
+ setDuration(int): void
+ setAvailableSeats(int): void
+ print_Passenger(int): void
+ printFlight(): void
+ Remove_Passenger(Passenger, int): void
+ bookSeat(Passenger): void
+ bookSeat(): void
1
1
Plane
- id: int
- isInternational: bool
- flights: Flight[20]
- numFlights: int
+ Plane(int, bool)
+ addFlight(Flight): void
+ printSchedule(): void
+ bookSeat(int): void
+ convertToTime(string): time_t
Finance Department
- name : string
+ Finance_Department()
+ ~Finance_Department()
+ verify_Payment(const Passenger&, const string&): bool