User
`<<Abstract>>`
User

- email: String
- password: String

+ User(email: String, password: String)
+ logIn(otherEmail: String, otherPassword: String): boolean

FlightService
`<<Interface>>`
FlightService

+ getItinerary(origin: String, destination: List<Flight>, numOfTransfers: int): List<Itinerary> {exception ParseException}
+ sortItineraries(itineraries: List<Itinerary>, sortBy: String): List<Itinerary> {exception InvalidSortException}

Client
Client

- firstName: String
- lastName: String
- address: String
- creditCard: String
- creditExpiry: Date
- bookedItinerary: List<Itinerary>

+ Client(email: String, password: String, firstName: String, lastName: String, address: String, creditCard: String, creditExpiry: Date)
+ book(toBook: Itinerary): void
+ getBooked(): List<Itinerary>

Admin
Admin

+ Admin(email: String, password: String)
+ getClientEmail(client: Client): String
+ getClientPassword(client: Client): String
+ getClientName(client: Client): String
+ getClientAddress(client: Client): String
+ getClientCC(client: Client): String
+ getClientExpiry(client: Client): Date
+ setClientEmail(client: Client, setEmail: String): void
+ setClientPassword(client: Client, setPassword: String): void
+ setClientName(client: Client, setFirstName: String, setLastName: String): void
+ setClientAddress(client: Client, setAddress: String): void
+ setClientCC(client: Client, setCC: String): void
+ setClientExpiry(client: Client, expiry Date): void

FlightManager
FlightManager

- allFlights: Map<String, List<Flight>>
- date: DateFormat

+ FlightManager(flights: List<Flight>)

Flight
Flight

- flightNum: String
- arrivalTime: Calendar
- departureTime: Calendar
- origin: String
- destination: String
- airline: String
- availableSeats: int
- flightDuration: Duration
- cost: double

+ Flight(flightNum: String, arrivalDate: String, departureDate: String, origin: String, destination: String, airline: String, availableSeats: int, cost: double)
+ timeBetweenFlights(flight: Flight): Duration

Itinerary<Flight>
Itinerary<Flight>

- List<Flight>
- totalCost: double
- totalTime: Duration
- dateTime: DateFormat

+ Itinerary()
+ addFlight(flight: Flight): boolean
+ removeFlight(flight: Flight): boolean
+ getItineraryId(): int

Database
Database

- List<Flight>
- List<Client>
- List<Admin>

+ Database()
+readFlightTxt(flightDir: String): void
+readClientTxt(clientDir: String): void
+readAdminTxt(adminDir: String): void
+ addNewClient(client: Client): boolean
+ addNewAdmin(admin: Admin): boolean
+ addNewFlight(flight: Flight): boolean
+ addFlightsFromFile(file: String): boolean
+ updateClient(client: Client): boolean
+ updateAdmin(admin: Admin): boolean
+ updateFlight(flight: Flight): boolean

ValidDate
ValidDate

+ validate(date: String): boolean

InvalidSortException
InvalidSortException

+ InvalidSortException()
+ InvalidSortException(message: String)

Comparator<T>
`<<Interface>>`
Comparator<T>

+ Compare(obj1: T, obj2: T): int

Note:
This is a built-in java library.

ItineraryTimeComparator
ItineraryTimeComparator

+ ItineraryTimeComparator()

ItineraryCostComparator
ItineraryCostComparator

+ ItineraryCostComparator()