Design document:

Obj: process flight data and user requests. Manage flights based on requests. Calculate best routes between cities (cost and time in consideration). Read flight data and user requests from the files and provide best paths and output the results into a result file.

CityNode.java:

Purpose: Represent city with flights and serve as a node in the graph to connect other cities/nodes through flight paths

Used:
cityName: The name of the city the node represents.
flights: A list of flight objects representing all direct flights departing from this city.

Methods used:
- add a flight to the city's list of flights
- get information about available flights
- getting specific flight details.

FLight.java

Purpose: Serve as an individual flight between two cities.

Used:
DepartCity: As a source
Arrive City: As a destination
Cost: to calculate how expensive
Time: how long it would take

Methods used:
- Getting cost
- Getting time
- Location of the flight (where its at)

fPlan:

Purpose: manage flight routes between cities and to find possible flight paths from a specified origin to a destination

Used:
Cities: holds all cityNode objs representing each city in the network
Marked: temp list used for visited cities during the search for flight path

Important Methods used:

addFlight():

Gets or creates CityNode objects for both depart and arrive cities using the custCity method.
Creates two Flight objects: one for the journey from the departcity to the arrivecity and another for the return journey.
Adds these Flight objects to the corresponding cities flights list.

fLocate(String depart, String arrive, char orderBy, PrintWriter writer)

Locate CityNode object for the depart and arrive. Has error if none.
Calls fPaths to find all possible paths between the two cities.
Sorts these paths based on cost or time
Outputs up to three of the best routes to the writer

fPaths(CityNode curr, CityNode arriveCity, List<Flight> currPath, List<List<Flight>> sumPaths)

Recurvsively searches all paths from cur city to destination city
Mark city as visited
If current city is destination add current path to the list of finished paths.

Main.java

Purpose: handles files, flight requests

Uses:

CityNode and Flight
Read req.txt using fPlan and data.txt and outputs into output.txt

```
    Class FlightList
        Method addFlight(departureCity, arrivalCity, cost, time)
            Add flight to list

    Class LoadData
        Method loadData(filename)
            Read file line by line
            For each line
                Parse line into parts
                If line is valid
                    Add flight to FlightList using addFlight
                Else
                    Log error

        Method loadRequests(filename)
            Read file line by line
            For each line
                Parse line into parts
                If line is valid
                    Create RequestF object
                    Add to requests list
                Else
                    Log error

    Class FLocate
        Method findFlight(graphs, departCity, arriveCity)
            Implement flight search algorithm (e.g., BFS)
            Return list of paths

        Method displayFlights(paths, preference)
            Sort paths based on preference
            Display paths

    Class Main
        Create instances of FlightList and LoadData
        Call loadData to populate flight data
        Call loadRequests to gather flight requests
        For each request
            Use FLocate to find and display flights

End Program
```