

# INFO1113

---

# Assignment 1

---

Due: 23 April 2021, 11:59PM AEST

*This assignment is worth 12% of your final grade.*

## Task Description – Flight Scheduler

In this assignment, you will create a Flight Scheduler application in the Java programming language. The program will be a tool for airlines to use to schedule flights between different locations, producing timetable plans, and an easy way to check routing between cities on multiple flights. You must create at least three classes: FlightScheduler, Flight and Location, for which a scaffold and description have been provided to you. The FlightScheduler class will contain the main entry point of the application (static main function).

You are encouraged to ask questions on Ed under the assignments category if you are unsure of the specification – but staff members will not be able to do any coding or debugging in this assignment for you. As with any assignment, make sure that your work is your own, and do not share your code or solutions with other students.

## Working on your assignment

You can work on this assignment on your own computer or the lab machines. It is important that you continually back up your assignment files onto your own machine, external drives, and in the cloud.

You are encouraged to submit your assignment on Ed while you are in the process of completing it. By submitting you will obtain some feedback of your progress on the sample test cases provided.

## Implementation details

Write a program in Java to implement the Flight Scheduler application that accepts input from the user via standard input. The terminal interface allows the user to interact with the program, to give it input and receive output. The available commands are described below in the section 'Commands'.

There are three main classes you must implement, but you may also create more if you wish.

### FlightScheduler class

This class will contain the main entry point of your program (static main function) and store links to all the data relevant to the application. It will be a container for the flight schedule, which is made up of a list of Flights. It should also contain a list of Locations.

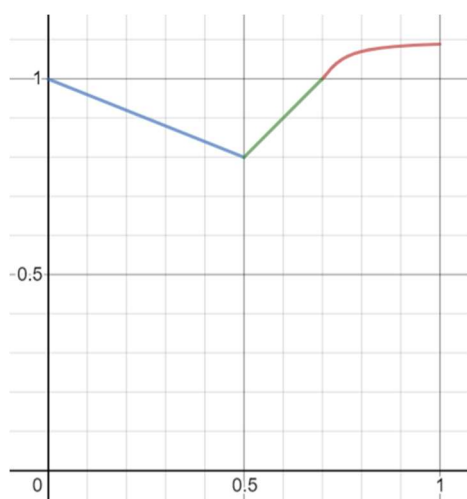
The flight schedule is only a single week, Monday to Sunday, which repeats. Assume all times are in UTC, so you do not have to account for timezone differences at different locations.

### Flight class

The Flight type should contain all data relevant to a particular flight, methods that perform operations on a Flight or multiple Flights. Attributes will be the flight ID, departure time, source and destination locations, capacity, ticket price, number of passengers booked, and anything else you think is relevant.

Flight duration is determined by the distance between the start and end locations, calculated using the [Haversine Formula](#), and assuming the average speed of an aircraft is 720km/h. The initial ticket price is calculated using an average cost of \$30, plus 4x the demand coefficient differential between locations, per 100km distance. For example, if the starting location has demand coefficient of -1 and the end has -1, it remains \$30 per 100km. If the starting location has -1 and the end has 1, then it's \$38 per 100km. If the starting location has 1 and end has -1, it would be \$22 per 100km.

Ticket price changes when the flight starts to fill up. For the first 50% of seats, the price decreases linearly to 80% of its original value by the time the flight is half full. For the next 20% of seats, the price increases linearly back to 100% of its original value. For the last 30% of seats, ticket price increases by an inverse-tan curve to 110% of its original value.



$$y = \begin{cases} -0.4x + 1, & 0 < x \leq 0.5 \\ x + 0.3, & 0.5 < x \leq 0.7 \\ \frac{0.2}{\pi} \times \tan^{-1}(20x - 14) + 1, & 0.7 < x \leq 1 \end{cases}$$

$$T = y \times \frac{d}{100} \times (30 + 4(D_{to} - D_{from}))$$

where

T = ticket price

y = multiplier for ticket price to determine current value

x = proportion of seats filled (booked/capacity)

d = flight distance in kilometres (haversine formula result)

$D_{to}$  = demand coefficient for destination location

$D_{from}$  = demand coefficient for starting location

### Location class

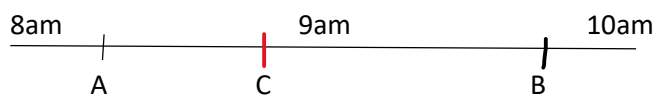
The Location type should contain all data relevant to a particular location, and methods that perform operations on a Location or multiple Locations. Attributes will be the location name, latitude and longitude coordinates, lists of arriving and departing flights, and a demand coefficient. Location names must be unique (case insensitive). Latitude must be within  $[-85, 85]$  and longitude must be within  $[-180, 180]$ , both in degrees. The demand coefficient is a number between -1 and 1 (inclusive) which represents whether there is a net inflow or outflow of passengers from this location (negative means passengers want to leave, positive means they want to come). It factors into the calculation that determines the ticket price for a particular flight.

Assume each location has only one runway – that is, no flights can be scheduled to arrive or depart within an hour of another at a particular location. Multi-runway airports can be represented by multiple locations in such a system (eg. Heathrow-1, Heathrow-2, etc).

If there is a scheduling conflict, you must check for conflicts in the following order:

- If there is a clashing flight departing within an hour of this flight's departure time, at the source
- If there is a clashing flight arriving within an hour of this flight's departure time, at the source
- If there is a clashing flight departing within an hour of this flight's arrival time, at the destination
- If there is a clashing flight arriving within an hour of this flight's arrival time, at the destination

If there are multiple conflicting flights, the message output should refer to the first one **immediately after** the flight attempting to be entered into the schedule for that location. If all of the conflicts **are before the** current flight, then the conflict output is the one immediately before the current flight in the schedule for that location. For example:



Flight C is attempting to be entered into the schedule at a particular location. There are flights A and B already in the schedule. Flight B is the one output as the conflict. If flight B was not present, flight A would be output as the conflict.

## Commands

FLIGHTS - list all available flights **ordered by departure time, then departure location name**

FLIGHT ADD <departure time> <from> <to> <capacity> - add a flight

FLIGHT IMPORT/EXPORT <filename> - import/export flights to csv file

FLIGHT <id> - view information about a flight (from->to, departure arrival times, current ticket price, capacity, passengers booked)

FLIGHT <id> BOOK <num> - book a certain number of passengers for the flight at the current ticket price, and then adjust the ticket price to reflect the reduced capacity remaining. If no number is given, book 1 passenger. If the given number of bookings is more than the remaining capacity, only accept bookings until the capacity is full.

FLIGHT <id> REMOVE - remove a flight from the schedule

FLIGHT <id> RESET - reset the number of passengers booked to 0, and the ticket price to its original state.

LOCATIONS - list all available locations in alphabetical order

LOCATION ADD <name> <lat> <long> <demand\_coefficient> - add a location

**LOCATION <name>** - view details about a location (it's name, coordinates, demand coefficient)

LOCATION IMPORT/EXPORT <filename> - import/export locations to csv file

SCHEDULE <location\_name> - list all departing and arriving flights, in order of the time they arrive/depart

DEPARTURES <location\_name> - list all departing flights, in order of departure time

ARRIVALS <location\_name> - list all arriving flights, in order of arrival time

TRAVEL <from> <to> [sort] [n] - list the nth possible flight route between a starting location and destination, with a maximum of 3 stopovers. Default ordering is for shortest overall duration. If n is not provided, display the first one in the order. If n is larger than the number of flights available, display the last one in the ordering.

can have other orderings:

TRAVEL <from> <to> cost - minimum current cost

TRAVEL <from> <to> duration - minimum total duration

TRAVEL <from> <to> stopovers - minimum stopovers

**TRAVEL <from> <to> layover - minimum layover time**

TRAVEL <from> <to> flight\_time - minimum flight time

HELP – outputs this help string.

EXIT – end the program.

---

**Note: All commands may be case insensitive.**

However Location names when stored in the location class, should display the name as initially given.

## Travel command

Since the schedule is weekly and wraps around, you need to consider the possibility of a flight arriving on Sunday evening potentially connecting with a flight that departs on Monday morning. As such, you may ignore available seat capacity selecting a flight in a potential route, since it is assumed that the current bookings are only for the current week, and this flight route may be used to show results for travellers in subsequent weeks, looking to make a booking later on. However, the ticket prices and overall route cost should depend on the **current booking numbers of each flight**, since we are assuming that the current booking demand is a good indicator of future demand, so ticket prices will be similar in the future to what they are now.

The TRAVEL command has 5 potential orderings, detailed below. If the primary sorting property is equal between two flight paths, it will fall back to the following secondary and tertiary sorting properties.

- If total cost is equal, sort then by minimum total duration.
- If total duration is equal, sort then by minimum current cost. Total duration is the time taken from initial departure of the first flight, to finally arriving at the destination.
- If number of stopovers is equal, sort then by minimum total duration (and then by minimum cost). Stopovers are intermediary locations travelled to in order to reach the destination.
- If layover time is equal, sort then by minimum total duration (and then by minimum cost). Layover time is the time spent waiting at the airport for connecting flights.
- If flight time is equal, sort then by minimum total duration (and then by minimum cost). Flight time is the time spent onboard the aircraft while it is flying (ie. total duration excluding layover time).

The output format of the travel command is composed of the flight plan, with layover times between flights specified, see the examples section below.

**Note:** The number of stopovers is the number of intermediary destinations, not including the original starting location and final destination. It is equivalent to the number of flight legs minus 1.

**Also:** The nth flight in the order, starts from 0 being the first one.

## Error messages

The following messages should be output upon encountering the prescribed error case or condition. The order of precedence is as shown here, except 'not enough arguments' takes precedence:

command	Description of condition/error case	Message output
flight add	Time added was not in the correct format, for example "Monday 18:00"	Invalid departure time. Use the format <day_of_week> <hour:minute>, with 24h time.
	Starting location was not in the database.	Invalid starting location.
	Ending location was not in the database.	Invalid ending location.
	Capacity was not a positive integer.	Invalid positive integer capacity.
	The two locations entered were the same.	Source and destination cannot be the same place.
	No runways are available for this flight at the designated location at that time. <DateTime> is in the format as above, "Monday 18:00"	Scheduling conflict! This flight clashes with Flight <ID> departing from <Location> on <DateTime>.
	No runways are available for this flight at the designated location at that time. <DateTime> is in the format as above, "Monday 18:00"	Scheduling conflict! This flight clashes with Flight <ID> arriving at <Location> on <DateTime>.
	Not enough command arguments given.	Usage: FLIGHT ADD <departure time> <from> <to> <capacity>\nExample: FLIGHT ADD Monday 18:00 Sydney Melbourne 120
flight book	The number of passengers entered was not a valid positive integer.	Invalid number of passengers to book.
	If the capacity is full, print this after each attempt at booking that would otherwise exceed the capacity, or has filled it.	Flight is now full.
flight remove	Remove this flight – display the short departure time, eg. Mon 18:00, and then source → destination locations.	Removed Flight <ID>, <DateTime> <Location> --> <Location>, from the flight schedule.
flight reset	Display the short departure time, eg. Mon 18:00, and then source → destination locations.	Reset passengers booked to 0 for Flight <ID>, <DateTime> <Location> --> <Location>.
flight <id>	Invalid flight id is entered that is either not a number, or does not exist in the database.	Invalid Flight ID.
flight	no parameters given	Usage:\nFLIGHT <id> [BOOK/REMOVE/RESET] [num]\nFLIGHT ADD <departure time> <from> <to> <capacity>\nFLIGHT IMPORT/EXPORT <filename>
import	No filename is given, or the file doesn't exist.	Error reading file.
export	No filename is given, or the directory for this file doesn't exist.	Error writing file.

flights/ locations	No flights/locations are available – print this instead of the flight/location listing	(None)
location add	Location is already present in the database (case insensitive based on name)	This location already exists.
	Latitude exceeds bounds or is an invalid number.	Invalid latitude. It must be a number of degrees between -85 and +85.
	Longitude exceeds bounds or is an invalid number	Invalid longitude. It must be a number of degrees between -180 and +180.
	Demand coefficient exceeds bounds or is an invalid number.	Invalid demand coefficient. It must be a number between -1 and +1.
	Not enough command arguments given.	Usage: LOCATION ADD <name> <lat> <long> <demand_coefficient>\nExample: LOCATION ADD Sydney -33.847927 150.651786 0.2
location	Location is not present in the database (case insensitive based on name)	Invalid location name.
location	No parameters given.	Usage:\nLOCATION <name>\nLOCATION ADD <name> <latitude> <longitude> <demand_coefficient>\nLOCATION IMPORT/EXPORT <filename>
Schedule, departures, arrivals	Location is not present in the database (case insensitive based on name)	This location does not exist in the system.
travel	Starting location is not present in the database (case insensitive based on name.	Starting location not found.
	Ending location is not present in the database (case insensitive based on name.	Ending location not found.
	Bad sorting property.	Invalid sorting property: must be either cost, duration, stopovers, layover, or flight_time.
	No flight paths of 3 stopovers or less are available from the given starting location to the ending destination.	Sorry, no flights with 3 or less stopovers are available from <Location> to <Location>.
	No parameters given.	Usage: TRAVEL <from> <to> [cost/duration/stopovers/layover/flight_time]

## CSV file formats

The import and export command for flights and locations allow the contents of the flight and location databases within the program to be saved to CSV (comma separated values) files. Two example files have been provided, as well as a sample command input/output sequence below.

When importing, if invalid lines are encountered in the file without the required data, skip them. Display the total number of invalid flights at the end, if any were invalid. An invalid flight is only considered if there were enough parameters in the line (ie. Exclude empty lines and lines without enough parameters from the invalid total). For example:

```
User: location import locations.csv
Imported 23 locations.
1 line was invalid.
```

```
User: flight import flights.csv
Imported 23 flights.
3 lines were invalid.
```

```
User: flight import flights2.csv
Imported 1 flight.
```

```
User: flight export flights3.csv
Exported 1 flight.
```

Flights and locations are to be imported in the order they are given in the file.

Export should write to the file flights in the order of their flight id, and locations in the order of their name.

Flights csv has the following format: day time,startLocation,endLocation,capacity,booked

Example:

```
Monday 18:00,Sydney,Melbourne,120,80
Monday 19:00,Sydney,Hobart,120,29
Monday 21:30,Sydney,Hobart,120,29
Monday 18:00,Auckland,Rio,120,1
```

Locations csv has the following format: locationName,latitude,longitude,demandCoefficient

Example:

```
Sydney,-33.847927,150.651786,0.4
Hobart,-42.8823399,147.3198016,0.1
Perth,-32.0397559,115.681346,0.5
Adelaide,-35.0004451,138.3309716,0.1
CoffsHarbour,-30.2973943,153.0286009,-0.2
Brisbane,-27.4732824,152.747337,0.3
```



## Examples – Input/Output format

Format of FLIGHTS command – sorted by departure time:

Flights

ID	Departure	Arrival	Source -->	Destination
0	Mon 17:00	Mon 17:04	Brisbane -->	GoldCoast
1	Mon 18:05	Mon 18:09	Brisbane -->	GoldCoast
2	Mon 19:05	Mon 19:09	Brisbane -->	GoldCoast

Format of flight <id> command:

Flight 0  
 Departure: Mon 00:05 Berlin  
 Arrival: Mon 08:57 NewYork  
 Distance: 6,387km  
 Duration: 8h 52m  
 Ticket Cost: \$1724.01  
 Passengers: 0/189

Format of locations command – sorted by alphabetical order:

Locations (3):  
 Berlin, London, NewYork

Format of location <name> command:

Location: Hobart  
 Latitude: -42.882340  
 Longitude: 147.319802  
 Demand: +0.5000

Format of travel command:

User: travel sydney london

Legs: 3  
 Total Duration: 34h 36m  
 Total Cost: \$5172.49

ID	Cost	Departure	Arrival	Source -->	Destination
5	\$ 3399.00	wed 10:00	Thu 02:44	Sydney -->	AbuDhabi
LAYOVER 1h 16m at AbuDhabi					
10	\$ 1384.44	Thu 04:00	Thu 11:11	AbuDhabi -->	Oslo
LAYOVER 7h 49m at Oslo					
20	\$ 389.05	Thu 19:00	Thu 20:36	Oslo -->	London

## Examples (1)

```
$ java FlightScheduler
```

```
User: location add Berlin 52.5 13.15 0.22222
```

```
Successfully added location Berlin.
```

```
User: location add NewYork 40.7 -74.26 -0.874
```

```
Successfully added location NewYork.
```

```
User: flight add sunday 20:00 Berlin NewYork 250
```

```
Successfully added Flight 0.
```

```
User: flights
```

```
Flights
```

```
-----  
ID    Departure    Arrival    Source --> Destination  
-----  
    0 Sun 20:00    Mon 04:52    Berlin --> NewYork  
-----
```

```
User: flight 0
```

```
Flight 0
```

```
Departure:    Sun 20:00 Berlin
```

```
Arrival:      Mon 04:52 NewYork
```

```
Distance:     6,387km
```

```
Duration:     8h 52m
```

```
Ticket Cost:  $1636.01
```

```
Passengers:   0/250
```

```
User: flight add monday 05:00 newYork berlin 234
```

```
Scheduling conflict! This flight clashes with Flight 0 arriving at  
NewYork on Monday 04:52.
```

```
User: exit
```

```
Application closed.
```

## Examples (2)

```
$ java FlightScheduler
User: flight import flights.csv
Imported 0 flights.
3 lines were invalid.
```

```
User: location import locations.csv
Imported 51 locations.
```

```
User: locations
Locations (51):
AbuDhabi, Adelaide, AliceSprings, Alta, Athens, Auckland, Beijing,
Berlin, Bern, Bordeaux, Brisbane, Cairo, Cardiff, Chicago,
CoffsHarbour, Dallas, Darwin, Dubai, Dubbo, GoldCoast, Hanoi, Hobart,
Houston, Jakarta, Johannesburg, Lagos, Liverpool, London, Longyearbyen,
LosAngeles, Luton, Madrid, Manchester, Moscow, NewYork, Orange, Oslo,
Paris, Perth, Rio, Rome, SanFrancisco, Stockholm, Sydney, Toulouse,
Townsville, Tromso, Ufa, Utqiagvik, Vladivostok, Washington
```

```
User: flights import flights.csv
Flights
```

```
-----
ID   Departure   Arrival      Source --> Destination
-----
```

```
(None)
```

```
User: flight import flights.csv
Imported 2 flights.
1 line was invalid.
```

```
User: flights
Flights
```

```
-----
ID   Departure   Arrival      Source --> Destination
-----
```

```
0 Mon 19:00   Mon 20:27   Sydney --> Hobart
1 Mon 21:30   Mon 22:57   Sydney --> Hobart
```

```
User: flight 0
Flight 0
```

```
Departure:    Mon 19:00 Sydney
Arrival:      Mon 20:27 Hobart
Distance:     1,045km
Duration:     1h 27m
Ticket Cost:  $272.00
Passengers:   29/120
```

```
User: flight 1
Flight 1
```

```
Departure:    Mon 21:30 Sydney
Arrival:      Mon 22:57 Hobart
Distance:     1,045km
Duration:     1h 27m
Ticket Cost:  $272.00
Passengers:   29/120
```

```
User: exit
Application closed.
```

## Examples (3)

User: location import locations.csv  
Imported 51 locations.

User: flight add wednesday 6:00 sydney perth 180  
Successfully added Flight 0.

User: flight add wednesday 8:00 sydney perth 180  
Successfully added Flight 1.

User: flight 0  
Flight 0  
Departure: wed 06:00 Sydney  
Arrival: wed 10:31 Perth  
Distance: 3,254km  
Duration: 4h 31m  
Ticket Cost: \$989.16  
Passengers: 0/180

User: flight 0 book 20  
Booked 20 passengers on flight 0 for a total cost of \$19365.60

User: flight 0 book 20  
Booked 20 passengers on flight 0 for a total cost of \$18486.35

User: flight 0 book 20  
Booked 20 passengers on flight 0 for a total cost of \$17607.09

User: fligh1 book 100  
Invalid command. Type 'help' for a list of commands.

User: fligh 1 book 100  
Invalid command. Type 'help' for a list of commands.

User: flight 1 book 100  
Booked 100 passengers on flight 1 for a total cost of \$88381.66

User: flight 0  
Flight 0  
Departure: wed 06:00 Sydney  
Arrival: wed 10:31 Perth  
Distance: 3,254km  
Duration: 4h 31m  
Ticket Cost: \$857.27  
Passengers: 60/180

User: flight 1  
Flight 1  
Departure: wed 08:00 Sydney  
Arrival: wed 12:31 Perth  
Distance: 3,254km  
Duration: 4h 31m  
Ticket Cost: \$846.28  
Passengers: 100/180

User: flight add wednesday 11:31 perth johannesburg 230  
Successfully added Flight 2.

User: fligh 2  
Invalid command. Type 'help' for a list of commands.

User: flight 2

Flight 2

Departure: Wed 11:31 Perth  
 Arrival: Wed 23:03 Johannesburg  
 Distance: 8,303km  
 Duration: 11h 32m  
 Ticket Cost: \$2163.44  
 Passengers: 0/230

User: flight add thursday 01:00 johannesburg London 220  
 Successfully added Flight 3.

User: flight add friday 01:00 johannesburg London 220  
 Successfully added Flight 4.

User: flight add saturday 01:00 johannesburg London 220  
 Invalid departure time. Use the format <day\_of\_week> <hour:minute>, with 24h time.

User: flight add saturday 01:00 johannesburg London 220  
 Successfully added Flight 5.

User: flights

Flights

ID	Departure	Arrival	Source --> Destination
0	wed 06:00	wed 10:31	Sydney --> Perth
1	wed 08:00	wed 12:31	Sydney --> Perth
2	wed 11:31	wed 23:03	Perth --> Johannesburg
3	Thu 01:00	Thu 13:36	Johannesburg --> London
4	Fri 01:00	Fri 13:36	Johannesburg --> London
5	Sat 01:00	Sat 13:36	Johannesburg --> London

User: flight export flights3.csv  
 Exported 6 flights.

User: exit  
 Application closed.

## Examples (4)

User: location import locations.csv  
 Imported 51 locations.

User: location import locations4.csv  
 Imported 2 locations.

User: flight import flights6.csv  
 Imported 182 flights.

User: travel sydney london

Legs: 3  
 Total Duration: 34h 36m  
 Total Cost: \$5172.49

ID	Cost	Departure	Arrival	Source --> Destination
----	------	-----------	---------	------------------------

```

-----
  5 $ 3399.00 Wed 10:00 Thu 02:44 Sydney --> AbuDhabi
LAYOVER 1h 16m at AbuDhabi
 10 $ 1384.44 Thu 04:00 Thu 11:11 AbuDhabi --> Oslo
LAYOVER 7h 49m at Oslo
 20 $ 389.05 Thu 19:00 Thu 20:36 Oslo --> London

```

User: exit  
Application closed.

## Examples (5)

User: location import locations.csv  
Imported 51 locations.

User: location import locations4.csv  
Imported 2 locations.

User: flight import flights6.csv  
Imported 182 flights.

User: **schedule sydney**  
Sydney

```

-----
ID   Time           Departure/Arrival to/from Location
-----
166 Mon 10:30      Departure to Dubai
100 Mon 12:00      Departure to Jakarta
103 Mon 13:33      Arrival from AbuDhabi
 89 Mon 19:00      Departure to Hobart
 90 Mon 21:30      Departure to Hobart
173 Tue 02:38      Arrival from AliceSprings
174 Tue 04:00      Departure to AliceSprings
 69 Tue 05:00      Departure to LosAngeles
 99 Tue 08:27      Arrival from Hobart
  0 Wed 09:00      Departure to Beijing
  5 Wed 10:00      Departure to AbuDhabi
161 Wed 11:00      Departure to LosAngeles
162 Wed 13:00      Departure to Dubbo
165 Wed 15:23      Arrival from Dubbo
163 Wed 18:00      Departure to Orange
164 Wed 20:14      Arrival from Orange
  6 Thu 04:44      Arrival from AbuDhabi
  3 Thu 11:11      Arrival from Hanoi
131 Fri 14:00      Departure to AbuDhabi
113 Sat 04:00      Departure to Perth
114 Sat 06:00      Departure to Perth
112 Sat 09:00      Departure to Perth

```

User: arrivals sydney  
Sydney

```

-----
ID   Time           Departure/Arrival to/from Location
-----
103 Mon 13:33      Arrival from AbuDhabi
173 Tue 02:38      Arrival from AliceSprings
 99 Tue 08:27      Arrival from Hobart
165 Wed 15:23      Arrival from Dubbo
164 Wed 20:14      Arrival from Orange
  6 Thu 04:44      Arrival from AbuDhabi

```

3 Thu 11:11 Arrival from Hanoi

User: departures sydney  
Sydney

ID	Time	Departure/Arrival to/from Location
166	Mon 10:30	Departure to Dubai
100	Mon 12:00	Departure to Jakarta
89	Mon 19:00	Departure to Hobart
90	Mon 21:30	Departure to Hobart
174	Tue 04:00	Departure to AliceSprings
69	Tue 05:00	Departure to LosAngeles
0	wed 09:00	Departure to Beijing
5	wed 10:00	Departure to AbuDhabi
161	wed 11:00	Departure to LosAngeles
162	wed 13:00	Departure to Dubbo
163	wed 18:00	Departure to Orange
131	Fri 14:00	Departure to AbuDhabi
113	Sat 04:00	Departure to Perth
114	Sat 06:00	Departure to Perth
112	Sat 09:00	Departure to Perth

User: arrivals perth  
Perth

ID	Time	Departure/Arrival to/from Location
169	Mon 17:52	Arrival from AliceSprings
113	Sat 08:31	Arrival from Sydney
114	Sat 10:31	Arrival from Sydney
112	Sat 13:31	Arrival from Sydney

User: departures perth  
Perth

ID	Time	Departure/Arrival to/from Location
145	Mon 00:00	Departure to Johannesburg
171	Mon 15:05	Departure to AliceSprings
170	Mon 16:05	Departure to AliceSprings
115	Thu 06:00	Departure to Mumbai
132	Fri 13:00	Departure to AbuDhabi

User: schedule perth  
Perth

ID	Time	Departure/Arrival to/from Location
145	Mon 00:00	Departure to Johannesburg
171	Mon 15:05	Departure to AliceSprings
170	Mon 16:05	Departure to AliceSprings
169	Mon 17:52	Arrival from AliceSprings
115	Thu 06:00	Departure to Mumbai
132	Fri 13:00	Departure to AbuDhabi
113	Sat 08:31	Arrival from Sydney
114	Sat 10:31	Arrival from Sydney
112	Sat 13:31	Arrival from Sydney

User: exit  
Application closed.

## Examples (6)

```
$ java FlightScheduler
```

```
User: location add Mumbai 19.08 72.741 0.4  
Successfully added location Mumbai.
```

```
User: location add NewDelhi 28.527 77.0688988 -0.123  
Successfully added location NewDelhi.
```

```
User: flight add monday 08:00 mumbai newdelhi 120  
Successfully added Flight 0.
```

```
User: flight 0
```

```
Flight 0
```

```
Departure:    Mon 08:00 Mumbai  
Arrival:      Mon 09:35 NewDelhi  
Distance:     1,139km  
Duration:     1h 35m  
Ticket Cost: $317.79  
Passengers:   0/120
```

```
User: flight 0 book 23
```

```
Booked 23 passengers on flight 0 for a total cost of $7041.08
```

```
User: flight 0 reset
```

```
Reset passengers booked to 0 for Flight 0, Mon 08:00 Mumbai -->  
NewDelhi.
```

```
User: flight 0
```

```
Flight 0
```

```
Departure:    Mon 08:00 Mumbai  
Arrival:      Mon 09:35 NewDelhi  
Distance:     1,139km  
Duration:     1h 35m  
Ticket Cost: $317.79  
Passengers:   0/120
```

```
User: flight 0 book 130
```

```
Booked 120 passengers on flight 0 for a total cost of $36255.03  
Flight is now full.
```

```
User: flight 0 remove
```

```
Removed Flight 0, Mon 08:00 Mumbai --> NewDelhi, from the flight  
schedule.
```

```
User: flights
```

```
Flights
```

```
-----  
ID    Departure  Arrival    Source --> Destination  
-----
```

```
(None)
```

```
User: exit
```

```
Application closed.
```



## Writing your own testcases

We have provided you with some test cases but these do not test all the functionality described in the assignment. It is important that you thoroughly test your code by writing your own test cases.

You should place all of your test cases in a tests/ directory. Ensure that each test case has a .in input file along with a corresponding .out output file. We require that the names of your test cases are descriptive so that you know what each is testing, e.g. listFlights.in & listFlights.out and we can accurately and quickly assess your test cases. **Note: If you do not format your test case files as explained (where each test case has <name>.in and <name>.out files for input and output, placed inside the tests folder), you shall receive 0 for this component.**

## Submission Details

You must submit your code and tests using the assignment page on Ed. To submit, simply place your files and folders into the workspace, click run to check your program works and then click submit.

You are encouraged to submit multiple times, but only your last submission will be considered.

## Marking

- 7 marks will be assigned based on the results of the automatic tests and correctness of the program. This component will use hidden test cases that cover every aspect of the specification. Your program must match the exact output in the examples and the test cases on Ed.
- 4 marks will be assigned to the code coverage of the testcases you have written yourself. For this, we will use a script to automatically generate a code coverage report using Jacoco. For this reason, please make sure you structure your testcases in the manner described above.
- 0.5 marks will be assigned based on a manual inspection of the OO design of your program. Being able to apply standard paradigms of Object-Oriented Design such as encapsulation, not repeating code, and separation of methods and attributes into different classes with a single responsibility is part of the learning outcomes of this course.
- 0.5 marks will be assigned based on a manual inspection of the code style. Style will be assessed based on the conventions set out in the Google Java Style Guide (<https://google.github.io/styleguide/javaguide.html> )

## Academic Declaration

By submitting this assignment you declare the following:

*I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.*

*I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgment from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.*

*I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.*

*I acknowledge that the School of Computer Science, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and/or communicate a copy of this assignment to a plagiarism checking service or in-house computer program, and that a copy of the assignment may be maintained by the service or the School of Computer Science for the purpose of future plagiarism checking.*