# CSC207H: Assignment 1

**Group size:** Individual

**Deadline:** Monday 25 September at 11 AM.

**Summary:** Write Flight, Airport, and FlightSchedule classes to pass a given set of unit tests.

## Learning objectives

* Use IntelliJ to develop Java classes
* Understand the basics of Java's type system and how to handle input and ouput
* Do test-driven development using Java's unit test framework (JUnit)
* Work with Java syntax

Javadoc and other documentation is not required for this assignment.

## Problem domain

Airplanes take off from an airport and land at a different airport during an individual leg of a flight. Flights can include multiple take-offs and landings, but never twice at the same airport during a single flight. Flights are identified by two capital letters (denoting the airline) and then three digits. For example, AC327 would be an Air Canada flight. Each airport is identified by a three-character code containing capital letters. For example, Toronto's Pearson International Airport is identified as YYZ. An airport should **NOT** have the same flight land there twice in the same day.

## Starter code and setting up

We provide a set of unit tests in the FlightTest class and nearly-empty classes for Flight and Airport. Each Flight has a list of airports at which it has landed at, and each Airport has a list of flights which have landed there.

As always, proper capitalization is vital.

You will write class FlightSchedule which will contain your main method. First it will take a file called FlightList.txt as input. FlightList will contain one line for each flight. Each line will be formatted like this:

FlightName dd-mm-yyyy | airport1 | airport2 | ... | airportn |

Where the number of airports n can be different for each flight. For example, one line in the file could look like:

AC123 25-09-2017 | YYZ | YVR | PVG |

Then, your main method will print to the screen: Enter the name of an airport. If the user types in the name of a airport that appeared in FlightList.txt, for example: PVG, the output to the screen will look like this:

PVG (AC123, WS592, AC225)

If the user types the word exit, your program should terminate. Otherwise, the program will continue to prompt the user for more airports. If the user types in a string that is neither exit nor the name of an airport on the list, the program will print to screen This is not a valid airport.

Here are the instructions for getting started:

1. Follow the instructions in Lab 1 to create a new project in IntelliJ called a1. When you are asked if you want to open the project in a new window, say yes.
2. Download [starter.zip](http://www.teach.cs.toronto.edu/~csc207h/fall/assignments/a1/starter.zip). Unzip it and move each file into the src folder of your project.
3. You should now see Flight, Airport, FlightSchedule and the test file under src on the left panel of your IntelliJ window. You will write the code for the missing methods in Flight and Airport. They are:

Class Airport: wasVisitedBy, onSameFlight, addFlight, equals, getName, and toString.

Class Flight: addAirport, equals, getAirports, getName, getDate, and toString.

You will also write the code for your main method in FlightSchedule.

You will also have to add JUnit to your project. This looks different on different operating systems. You can find and share advice for accomplishing this on the discussion board or drop by office hours.

Your job for this assignment is to write those methods and make the test cases pass. Then write the main method. If you pass all of the test cases **on the teaching lab computers** and handle a test file of the specified format, you will earn 100% on the assignment.

## Equality

Two Flight objects are equal if they have the same name and date.

Two Airport objects are equal if they have the same name and have been visited by the same flights.

## A suggestion for doing this assignment

First, create method stubs so that all the unit tests compile. A method stub is a method header and just enough of a body to get it to compile.

For example, you might write these stubs for getName and wasVisitedBy in class Airport:

public String getName() {

return null;

}

public boolean wasVisitedBy(Flight f) {

return false;

}

Once your code compiles, run the tests. Many will fail, but your setup phase is done!

Now pick a unit test and read it carefully. We recommend starting with testFlightGetName because it's first, but you're welcome to start elsewhere. Write just enough code for the unit test to pass. Move on to the next unit test and do the same thing: read it carefully so that you understand what it is testing, and then write just enough code for it to pass.

You will sometimes need to iterate over part of the contents of an ArrayList. You can use a loop that looks like this one, which iterates over all the items in the list:

int i = 0;

while (i != this.airports.size()) {

// call this.airports.get(i), cast it to type Airport, and do something with it.

i = i + 1;

}

You can, of course, also use a for loop.

## Line separators

Unix, Windows, and Mac all use different newline separators. Linux and MacOS use \n, but Windows uses \r\n. System.lineSeparator() returns the appropriate one, depending on the operating system on which the code is executed. You **must** use it to indicate newlines; this is relevant to method toString in class Flight.

## Handling Input

You will want to explore the split method in class String when writing your main method. This method takes a regex as an argument. Regex is short for Regular Expression. We will be learning about these later in the course. For now, you will need to know that "\s" is the regex for a single blank space and "\\|" is the regex for the symbol | in Java.

There is a file called ReadWrite.java in the Week 2 Readings on the course website. You can re-use any of the code in this file, if appropriate. It demonstrates various methods for inputting text into your program. You are responsible for looking up unknown methods or classes on PCRS, the Oracle website, or from reliable alternative sources.

## Submitting the assignment

You will submit the assigment on MarkUs. The link for this is on the course website. Once there, log in with your UTORid and click on Assignment 1. Try to do this before the actual deadline so that you can seek assistance if required.

## Re-mark Requests

Your code for a1 must run on the teaching lab servers. Once the marks for a1 have been released on MarkUs, we will send out an e-mail regarding re-marking interviews. In order to get your assignment re-marked, you will make an appointment to meet with a member of CSC207 teaching team in a teaching lab where you can log in and show us that your code passes all of the tests. It is unlikely that you will need to do this, since your assignment will be graded on the teaching server. So we will see the same results that you see when you run your assignment on IntelliJ on a computer in a teaching lab.