Assessed Coursework 2 — Flight Itinerary

This exercise should be done **individually**. The deadline for this coursework is 3:30pm on the Wednesday of week 11. You should submit both a hardcopy and an electronic version. Parts of the coursework should be demonstrated in the lab. We recommend that you tackle each part of the coursework in turn over a number of weeks to spread the load. The demonstration of all parts must be complete by the last lab session you attend in week 11. All code must be submitted on Vision by Wednesday, November, 26$^{\texttt{th}}$, 3.30pm. This coursework is worth 50% of the total coursework mark for this module.

Description

Your task is to use a graph to represent airline data, and to support searching. You should carefully test all of the code you write, generating new data files as necessary, and include the result of your tests in the report.
The coursework aims to reinforce your understanding of module material, specifically the following learning objectives:

- Gain an understanding of a range of graph classes and their use to represent realistic data.
- Gain further experience in object-oriented software engineering with a non-trivial class hierarchy: specifically selecting an appropriate class; reusing existing classes; extending existing classes.
- Using generic code: reusing generic graph classes, and parameterising a class with different types.
- You will also gain general software engineering experience, specifically downloading and using Open Source software, using a general method for a specific purpose, and issues with reusing existing code.
- Gain further experience with Java programming.

Preliminary Part: Installation and get familiar with JGraphT

You will need to download a personal copy of the Open Source JGraphT graph library. Having a single central copy of the library for all students on the course would save space, but it is a valuable experience for you to use an Open Source code repository.

Graph Package Download & Setup

- Download the Open Source JGraphT graph library by following the instructions on:
  http://jgrapht.org/
  The following instructions are for `jgrapht-0.9.0` on a Unix machine using bash.
- Decompress and extract the tarball (this will create a 15M `jgrapht-0.9.0` directory), e.g.

```
linux22% tar zxvf jgrapht-0.9.0.tar.gz
```

- Delete the tarball to save 3.6Mb! (4.6M for the zip file), e.g.

```
linux22% rm jgrapht-0.9.0.tar.gz
```

- Add JGraphT demo and jar files to your class path at the end of your `.profile` file in your home directory.

  In the following command you should replace `yourlogin` with *your login*, and `yourpath` with the *directory path* to where you have installed JGraphT. That is,

  ```
  export CLASSPATH=/u1/cs2/yourlogin/yourpath/jgrapht-0.9.0/lib/↵
     jgrapht-core-0.9.0.jar:/u1/cs2/yourlogin/yourpath/jgrapht↵
     -0.9.0/source/jgrapht-demo/src/main/java/org/jgrapht/demo↵
     :.:$CLASSPATH
  ```

  For Jane Doe whose login is `jd42` and who is working on the assignment in the directory `F28DA/CW2`, the command is:

  ```
  export CLASSPATH=/u1/cs2/jd42/F28DA/CW2/jgrapht-0.9.0/lib/↵
     jgrapht-core-0.9.0.jar:/u1/cs2/jd42/F28DA/CW2/jgrapht↵
     -0.9.0/source/jgrapht-demo/src/main/java/org/jgrapht/demo↵
     :.:$CLASSPATH
  ```

- Execute your new `.profile`, e.g.

  ```
  linux22% source ~/.profile
  ```

Demonstration Programs

Go to the JGraphT source directory, e.g.

```
linux22% cd ~/yourpath/jgrapht-0.9.0
linux22% cd source/jgrapht-demo/src/main/java
```

Compile and execute the 1st demo program:

```
linux22% javac org/jgrapht/demo/HelloJGraphT.java
linux22% java org/jgrapht/demo/HelloJGraphT
```

Execute other demo programs, e.g. `PerformanceDemo` - takes several minutes!

View Javadoc

You will need to use the JGraphT Javadoc to locate appropriate classes and methods. Browse
http://jgrapht.org/javadoc/

Editing & Compiling Graph Programs

To edit a program

```
linux22% cd ~/yourpath/jgrapht-0.9.0
linux22% cd source/jgrapht-demo/src/main/java
```

Edit `HelloJGraphT.java` (`org/jgrapht/demo/HelloJGraphT.java`), by adding a new edge to one of the graphs. Compile and run your revised program

```
linux22% javac org/jgrapht/demo/HelloJGraphT.java
linux22% java org/jgrapht/demo/HelloJGraphT
```

Congratulations, you are ready to start writing graph programs.

Viewing Example Graph Programs

Often the easiest way to write a program is to reengineer, i.e. copy and modify, a similar program. More example programs are available in the test directory

```
linux22% cd ~/yourpath/jgrapht-0.9.0
linux22% cd source/jgrapht-core/src/test/java
```

[0]

Demonstration: Editing & Compiling must be demonstrated during week 9, and you should aim to complete Part A.

Part A: Representing Direct Flights

Write a program to represent the following direct flights with associated costs as a graph. For the purpose of this exercise assume that flights operate in both directions with the same cost, e.g. Edinburgh ↔ Heathrow denotes a pair of flights, one from Edinburgh to Heathrow, and another from Heathrow to Edinburgh.
Hint: Flights are directed, i.e. from one airport to another, and weighted by the ticket cost, hence use the JGraphT `SimpleDirectedWeightedGraph` class. You should display the contents of the graph (and may omit the weights).

Flight	Cost
Edinburgh ↔ Heathrow	£110
Heathrow ↔ Amsterdam	£100
Heathrow ↔ Boston	£230
Boston ↔ Chicago	£150
Boston ↔ Montreal	£100
Montreal ↔ Toronto	£90
Edinburgh ↔ Chicago	£560
New Delhi ↔ Shanghai	£430
Shanghai ↔ Hong Kong	£230

[10]

Part B: Least Cost Connections

Extend your program to search the flights graph to find the least cost route between two cities consisting of one or more direct flights.
Hint: use methods from the `DijkstraShortestPath` class to find the route. A possible interface for your program might be one where you suggest a start and an end city and the cost of the entire route is added up and printed.

```
The following airports are used:
       Edinburgh
       Heathrow
       ...
 Please enter the start airport
    Edinburgh
 Please enter the destination airport
    Amsterdam
 Shortest (i.e. cheapest) path:
 1. Edinburgh -> Heathrow
 2. Heathrow -> Amsterdam
 Cost of shortest (i.e. cheapest) path = £210
```

[15]

Demonstration: Part A and B must be demonstrated during week 10, and you should aim to complete at least Part C.

Part C: Additional Flight Information

Start a **new program** operating on a graph containing the same flights as in part A, but including the following information about each flight. The flight number, e.g. BA345; the departure time; the arrival time; the flight duration; and the ticket price, e.g. £100. All times should be recorded in 24 hour hhmm format, e.g. 1830.
Use your imagination to populate your graph with sensible flights.

[7]

Part D: Itinerary

Use the additional flight information to print itineraries for least cost journeys in a format similar to the following example. The key aspects are

1. A sequence of connecting flights (with least cost)
2. A total cost for the route

An example itinerary for parts E & F might resemble:

```
Itinerary for Edinburgh to Toronto
Leg Leave     At   On    Arrive   At
1   Edinburgh 1030 BA345 Heathrow 1130
2   Heathrow  1400 BA657 Boston   1530
3   Boston    1800 AA652 Montreal 1930
4   Montreal  2200 AA216 Toronto  2330
Total Journey Cost    = £510
Total Time in the Air = 930hrs
```

[15]

Demonstration: Part C and D must be demonstrated during week 11, and you should aim to complete at least Part E.

Part E: Itinerary Duration

Extend your program to calculate the total time in the air, i.e. the sum of the durations of all flights in the itinerary.
Hint: you will need to write functions to perform arithmetic on 24 hour clock times.

[8]

Part F: Alternative Extensions

Attempt at least three of the following extensions to your second program.

1. Airline itineraries record arrival and departure times in local time for the destination airport, also in 24 hour format. Calculate the total journey time as the sum of the flight durations plus the sum of the changeovers.

2. Extend your program to calculate journey time for journeys that span more than one day, e.g. takeoff at 21:30 and arrive at 04:00.

3. Extend your program to allow connections between two flights only if the arrival time of the first flight is between 1 and 5 hours of the departure time of the second flight.

4. Extend your program to locate routes with the fewest number of changeovers.

   Hint: use a standard graph traversal algorithm, available in JGraphT.

5. Extend your program in some other way. Carefully describe the extension in your report.

Your report must clearly indicate which extensions you implemented and must demonstrate them.

[15]

General

Program structure, comments and indentation: **[10]**
Demonstrations: **[15]**
Report with appropriate length, clear structure, lack of spelling and grammatical mistakes. **[5]**

Total Marks: 100

Submission Requirements

Submit both a hardcopy of your coursework in the coursework box (printed report) and an electronic version on Vision (Java code and report).

Hardcopy submission The hardcopy should be a brief report (max. 7 pages, excluding the appendices), prepared using the word processor of your choice, and containing the following sections.

Section 1 Testing

This section presents a description of test data and testing outcomes, and includes reasons why test data was chosen.

1.1 Program 1, Part A: Representing Direct Flights
1.2 Program 1, Part B: Least Cost Connections
1.3 Program 2, Part C: Additional Flight Information
1.4 Program 2, Part D: Itinerary
1.5 Program 2, Part E: Itinerary Duration
1.6 Program 3, Part F: Alternative Extensions

In subsection 1.6 you should identify which extensions you have implemented, and give a demonstration of each.

Section 2 Evaluation

As part of your submission you should include a reflection on your coursework (max. 1 page). Please make it clear if parts of your code do not fully work. If some of your work falls into this category then include some analysis of the problem and how you would tackle it given more time.

Deadline: 3:30PM on Wednesday 26th of November, 2014

Electronic submission The electronic version consists of all the files of code you have written and the electronic version of your brief report. Your code should contain comments that clearly identify parts A, B etc. For example:

```
*****************************************************
* Part B: Least Cost Connections
*****************************************************
```