CSC207 Introduction to Software Design
Fall 2015 – Project Phase III

Logistics

- **Preliminary interviews (worth 3%):** week of 30 November 2015
- **Due date:** 9:00pm Tuesday 08 December 2015
- **Team evaluation due date:** 11:59pm Thursday 10 December 2015
- **Group size:** Normally four. In this phase of the project you continue to work in your team from Phase II.

Overview

In Phase III of the project you will complete the implementation of the Android application for searching air travel itineraries.

Learning Goals

By the end of this phase, you should have:

- practised dealing with software requirements that change over the course of a software development project,
- worked closely with your teammates to re-evaluate and possibly update your design of a software system,
- produced a working Android application that implements your software design and corresponds to user requirements.

Phase IIIa Interview (week of 30 November 2015)

The purpose of the interview is to demonstrate to your TA that you have a working (although very limited!) Android application. The application must be able to do the following:

1. Launch and start the main activity.
2. Take some input from the user and use it (in a `.java` class).
3. Transition from one `Activity` to another `Activity` and carry information between the two `Activity`s in an `Intent`.
4. Use at least one class from your back-end (implemented in Phase II of the project) intelligently.

All team members must be present for the interview and should be prepared to answer questions about the implementation of the Android application. Your TA will contact you to schedule the meeting.

Feature List for this Phase

Here are the features that you will implement for this Phase of the project. A lot of this functionality you should have already implemented in the previous phase of the project.

- A User (Client or Admin) can launch the flight booking application and login using a username and password; this loads saved data, if it exists. (In our unrealistic implementation, you are allowed to simply store usernames and passwords in a file on the device[1].)
- A User (Client or Admin) can search available flights by entering a departure date, and travel origin and destination. Each matching flight should be reported with: (1) flight number, (2) departure date and time, (3) arrival date and time, (4) airline, (5) origin, (6) destination, (7) cost, and (8) travel time. The flight numbers should be unique. We will not deal with time zones, leap years, or any calendar or timing differences, in this project.
- A User (Client or Admin) can search available itineraries by entering a departure date, and travel origin and destination. Each matching flight should be reported with, per flight: (1) flight number, (2) departure date and time, (3) arrival date and time, (4) airline, (5) origin, and (6) destination, plus an overall itinerary cost and travel time.
  A valid itinerary contains no stopvers over 6 hours long, and no cycles, i.e. it does not visit the same place more than once.
- A User (Client or Admin) can display search results sorted by total travel time or by total cost.
- A Client can view personal and billing information stored for that Client.
- An Admin can view personal and billing information stored for any Client.
- An Admin can upload (in a csv file – see format below) personal and billing client information into the system. If there is a subsequent upload, it should add to the existing data, rather than overwrite it. If the same client occurs again in a subsequent upload, their new data should replace the old data.
- A Client can edit personal and billing information for that Client.
- An Admin can edit personal and billing information for any Client.
- An Admin can upload (in a csv file – see format below) flight information into the system.
- An Admin can edit information for a flight.
- A User (Client or Admin) can select an itinerary from the displayed list for booking.
- A Client can book an itinerary for that Client.
- An Admin can book an itinerary for any Client.
- A Client can view booked itineraries for that Client.
- An Admin can view booked itineraries for any Client.
- All information stored in the system should persist (be available in the next launch) when the application is not running.
- To avoid loss of data, for example if the app crashes, data should be saved after every change, such as an upload or edit operation.

The user will use a `csv` file to upload personal and billing information about clients. The file format is as follows:

```
LastName,FirstNames,Email,Address,CreditCardNumber,ExpiryDate
```

where the expiry date is stored in the format `YYYY-MM-DD`. Flight information will also be uploaded using a `csv` file. The file format is as follows:

```
Number,DepartureDateTime,ArrivalDateTime,Airline,Origin,Destination,Price,NumSeats
```

where the date and time is stored in the format `YYYY-MM-DD HH:MM`. For both types of input file, there are no commas other than between fields, as shown above.

[1]Requirements for passwords file:

1. In the PIII directory, commit the plaintext password file, `passwords.txt`, that you will be using for authentication.
2. In your app, use the default internal storage location for storing the password file:
   `this.getApplicationContext().getFilesDir()`

Our airlines are not very client friendly — they do not assign seats at the time of booking. But they don't overbook either. They simply sell up to the available number of seats on the flight, which is specified by the `NumSeats` value in the flights data.

Task 1 — Software Design

With your teammates, discuss what changes you need to make to your design from Phase II of the project to handle the revised requirements or to correct flaws in your earlier design. Create a file `crc.pdf`, following the same format you used in Phase II, and commit this file to the directory `PIII` of your team repository.

In contrast to Phase II of the project, in this Phase your CRC model should also include CRC cards for the front-end classes (the activities). Make your CRC cards match the final version of your code.

Task 2 — Implementing the Android Application

When setting up your Android Application Project, select:

- Minimum Required SDK: API 8: Android 2.2 (Froyo)

The software development process

Your team should meet regularly while working on the project. You are required to have two types of meetings — planning meetings and status meetings.

You need to have **two planning meetings**: one in the beginning of this phase of the project and one mid-way through the project phase. During a planning meeting, the team will (a) recap on the current state of the project (if mid-way meeting), (b) decide on a set of tasks the team will accomplish before the next planning meeting, and (c) decide who will perform which tasks.

In addition to the planning meetings, the team will meet for **weekly status meetings**. (You may have more frequent status meetings if you wish.) During status meetings, each member will report on (a) what (s)he has accomplished since the last meeting, (b) what (s)he plans to accomplish before the next meeting, and (c) if there are any problems or obstacles that prevent him/her from making progress.

To demonstrate the software development process the team followed, you need to **maintain a plain text file** called `meetings.txt`, where the team will record all meeting minutes. (See the lecture slides for some example meeting minutes.) *On the day of each meeting*, commit this file into the directory for this phase in your team repository. The contents of this file must match the state of the rest of your repository!

If a member repeatedly does not meet deadlines or misses meetings, contact your instructor immediately.

The end of this project phase

At the end of this project phase, your team should have a working version of an Android application that implements every feature on the above feature list. You should, of course, have Javadoc comments for all your code.

Class `Driver`

One of the tasks in this phase is to complete your back-end implementation. Since the object-oriented design is up to you, every group is likely to have a different design. So that we can test your code, you must submit (in addition to your code that implements the features above) a class named `Driver` that implements the methods specified in the `Driver.java` starter code. This provides us with an API that is common for all teams, and that we can use to test your back-end code.

Notice that class `Driver` is required for testing purposes only! It should not contain any algorithms or do any interesting work. It should merely call appropriate methods in the classes that you designed and implemented.

`Driver.java` must be in package `driver`. Do not change the method signatures in class `Driver`.

README.txt

Please include a plain text file **README.txt** in directory PIII of your repository telling your TA where to find your Phase III project. You must tell your TA everything they need to know in order to run your application and navigate your work. If it is necessary to push particular files to the emulator, those instructions should be included here.

Your TA has a fixed amount of time to mark your app. Make sure that your **README.txt** file let's them convince themselves that your app works.

Task 3 — Team member and self evaluations

Any student who does not submit their evaluations on time will receive a mark of 0 on Phase III of the project.

You will be filling out and submitting a peer evaluation activity on CATME. This form will rate all team members, including yourself, on contributing to the team's work (contributing a sufficient amount of work, contributing work of good quality, being on time, helping teammates) and interacting with teammates (showing interest in teammates' ideas and contributions, asking teammates for feedback and using their suggestions to improve, making sure teammates stay informed and understand each other, providing encouragement and enthusiasm to the team).

These are meant to be private: each team member will submit these separately, and you are not required to show each other your forms. In the case of serious disagreement, or if you request it, we will hold a team meeting to discuss the results, but we will never reveal individual ratings.

Although your formal evaluations are private, that doesn't mean you shouldn't talk with your team members about how the team is functioning. It's much better to be open about this. In our experience, teams who ignore problems and just carry on with the work have the worst results.

Marking

All of these items may affect your grade:

- CRC Model
  - The modularity of the design, and the degree to which it is reusable and extensible.
  - The degree to which the design meets the requirements.
  - The use of OO concepts, such as encapsulation and inheritance.
  - How well the design matches your code.
- The appropriate use of files and data structures:
  - Have you made reasonable choices from among the many possibilities?
- Functionality and usability of the application:
  - all functions from the feature list implemented
  - stability of application (e.g., it shouldn't crash on invalid input)
  - easy to use application, intuitive navigation
- Javadoc:
  - required for methods and instance and static variables
  - must have a period at the end of every sentence
  - must use @param and @return tags
  - must use good English

- Coding Style:
  - must follow Java naming conventions
  - indentation
  - consistency
  - white space
- Quality of the software development process:
  - the file `meetings.txt` must be committed according to the schedule
  - the contents of the repository and the state of the code must match the contents of the file `meetings.txt`
- Subversion commit history:
  - participation by all team members
  - frequent commits over an extended period of time
  - appropriate commit logs
- Peer evaluation
  - To view the evaluation criterion, see the CATME online evaluation form (`www.catme.org`).
- Quality of the `README` file:
  - it must take the TA less than 2 minutes to read your `README` file and understand how to run and use your application

Checklist

Have you…

- used your new team repository and submitted your work to directory PIII?
- committed `crc.pdf`?
- committed **all** of the necessary Android files? (Everything except for the contents of the bin directory, the gen directory, and hidden files.)
- committed `Driver.java` in a package named `driver` in your project?
- committed a `README.txt` file for your TA?
- committed `meetings.txt`?
- verified that your changes were committed using `svn list` and `svn status`?
- before the separate deadline for team evaluations: submitted your team evaluation forms using CATME?