COSC1290/1295 Java for (C) Programmers
School of CS and IT, RMIT University
Assignment 2 – Semester 2 2016

Due date: 9.30pm, October 13, 2016

Introduction

You are required to implement a basic Java program using Java SE 8.0 and JavaFX.
This assignment is designed to help you:

1. Enhance your ability to build a Graphical User Interface using JavaFX;
2. Practise implementation of various GUI event handlers;
3. Practise application of using file input and output processing;
4. Practise the use of command-line arguments.

This is an individual assignment and worth **10%** towards your final grade.

Academic Integrity (more)

The submitted assignment must be your own work. No marks will be awarded for any parts which are not created by you---**this includes use of 3rd-party GUI components**.
Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files; this may be regarded as plagiarism. You should always create your own assignment even if you have similar ideas. Plagiarism-detection tools will be used for all submissions. Severe penalties may be applied in cases of plagiarism, including loss of all marks or, in repeat cases, expulsion from the course.

General Implementation Details

- Initial data should be read from an input text file. A Save option should be implemented that writes out to a text file.
- All menu/interactive input should be via a graphical user interface, implemented in JavaFX. All information displayed to the user should be via the same GUI.
- Sample GUI layouts will be put on Blackboard: **you are free to have different layouts as long as your GUI layouts are clear** (marks may be deducted for very poor design).
- Any incorrectly formatted input should be checked by the GUI implementation.
- You are to use packages: all your new code should be in packages called:
    - gui
    - fileio

  Code not in either of these packages will not be considered for marking.
- **You must not use 3rd-party GUI components (i.e., not built by you).**
- Marks will be allocated to proper documentation and coding layout and style. Your coding style should be consistent with standard coding conventions (as in Assignment 1).
- Your programs will be marked on CSIT machines using Java SE 8.0. You should test your program on these machines before you submit.

Task Specification

You are to build a Graphical User Interface (GUI) for the MyTi system of Assignment 1. You will also make running of the MyTi system more configurable by reading initial Card, User and TravelPass information from input files and write updated information to an output file. You do not need to add any other new functionality to the MyTi system of Assignment 1.

What if my Assignment 1 did not work properly?

- If you did not get TravelPass purchasing and Journey-taking working fully in Assignment 1 then you are allowed to just purchase a TravelPass (as in the first Sample Run).

Part A: Packages and Organisation of Code (0.5 marks)

You should organise your code into 3 packages. A package called **`myti`** should be used for code from Assignment 1. Packages called **`gui`** and **`fileio`** should contain all new code for Assignment 2 (you can put some of your existing code into these packages, if necessary).

Part B: Command-Line Arguments (0.5 mark)

You are to use command-line arguments to provide the name of your input and output files. For example, if your main() program is in the class `Assign3` and you read input from `input.txt` and your output is `output.txt`, then you would run your program as follows from the command-line on a unix machine:

> java Assign2 input.txt output.txt

If you use eclipse, then you specify the arguments under Run > Run Configurations; however, as always, your program will be marked by running it from the command-line.

Part C: File Input and Output (1.5 marks)

Put initial TravelPass prices and Card andUser information in a text file (the name of this file is the first command-line argument); your program should read this file at start-up and initialise prices and create Users with the appropriate values. A sample input file will be provided on Blackboard: you must follow that format precisely, though you are welcome to include more information than in that sample.

Your program will be tested with different values (but with the same format) so make sure you can handle reading the file properly.

When the SAVE option is selected on the GUI, you should write an output (text) file. The output file must use the identical format to input file format below. You should write out the same information as you read in with the updated values.

Part D: Graphical User Interface (6 marks)

Add a GUI for at least the following functionalities from Assignment 1. (You can implement a GUI that covers your complete Assignment 1 functionality but there's no extra marks in it.)

- The GUI should begin with a single window for purchasing Travel Passes and taking Journeys. Buttons for "Managing Cards and Users" and for "Display Reports" may open up new windows. See the sample GUI on Blackboard;
- The *Take a Journey* window should include:
    - a list for selecting a Card id (i.e., Take a Journey on this Card);
    - lists to select 2 stations;
    - a list to select the day of travel, and an input box to enter the time;
    - a message box where error messages or other output is displayed;
    - *CANCEL* and *PURCHASE* buttons;
- The *Manage Passes and Users* screen should include:
    - a list of Card IDs and associated User names;
    - when a Card is selected, User details and remaining credit is displayed;
    - an input box and an *ADD CREDIT* button for adding credit for the selected Card: **when this is performed, credit for that Card is displayed / updated**;
    - a message box where error messages or other output is displayed;
    - an *Add a User* panel that obtains the required information for a new user and adds them. When a User is created, there should be an option to register a Card to that User;
- The *Display Reports* screen should include:
    - a single Display All Journeys button;
    - a scrollable text box that contains the list of Journeys.
    - Note, if in Assignment 1 your function just displayed the Journeys for a Card, then you can do this here: you'll need a drop-down box to select the Card to display;
- SAVE and QUIT (without saving) buttons should always be available (ideally, outside the JTabbedPane).
- Each window will need buttons to open the other window options: for full marks, if a window is already open then it just shows it on top of the others, not generate a new one.

Obvious GUI errors should be checked—for example, if the user tries to *ADD CREDIT* without first selecting a Card from the menu.

Sample screens will be added to Blackboard. You do NOT have to follow these designs; they are simply a suggestion and you are free to design your own screens, as long as your GUI functionality is completely clear.

- **GUI Demo**: 1 / 0.5 marks of the GUI functionality will be for a demo in Week 10 / 11; this can be of "click" functionality that isn't fully connected to the MyTi system.

Part E: Other (1.5 marks)

As always, marks will be awarded for coding style, documentation/comments, code layout and clarity, meaningful error and other messages, proper error handling, choice of data structures and other design decisions. You are encouraged to discuss such issues with your tutors and lab assistants, or with the coding mentors.

Bonus Task (1 mark)

Instead of reading/writing from files, read/write from a database using JDBC.

Submission

Assignment submission will be via Blackboard, by 9.30pm, Thursday October 13 2016. You can submit your assignment as many times as you want before the due date. Each submission will overwrite any previous submissions.

1. You are encouraged to submit your .java files weekly via Blackboard. Your progress will be taken into consideration in marking. **There will also be a demo for this assignment**. You will be asked to demonstrate your progress in Weeks 10 and 11. Details of the demo will be announced on Blackboard and in lectures.
2. **You must include a README file.** This should describe how to run your program, what extra functionality you implemented, any standard functionality you know does **not** work, and any problems or assumptions. If the tutors have any problem running your program and the README does not help then **you will lose marks**.
3. For the code submission, you must include all **source files and input data file** in your submission (do not submit any *.class files!). Your code must run under yallara. **Since you are using packages, make sure you zip to the top-level of your package hierarchy: i.e. to the level at which you compile and run the main class.** (Ask your lab assistant to help if you are not sure how to do this.)
4. Late final submissions will incur a penalty of 10% per day. Submissions made 5 days after the due date will receive no marks.