COSC1295 Advanced Programming
Computer Science and Information Technology
School of Science, RMIT

Assignment 2 – Semester 1 2018

Due: 22 May 2018

Introduction

This assignment is worth 25% towards your final grade. You are required to implement a basic Java program using Java SE 8.0 or 9.0. This assignment is designed to:

- Practise your knowledge of exception in Java;
- Practise your knowledge of testing and debugging in Java;
- Practise the implementation of file handling, database handling in Java;
- Practise the implementation of Graphic User Interface in Java;
- Basic project management/development skills, e.g. using GitHub, collaboration and documentation.

This assignment can be submitted as a group of two (no more than two). We encourage group submission.

Academic Integrity

The submitted assignment must be your own work. No marks will be awarded for any parts which are not created by you. More on http://www.rmit.edu.au/academicintegrity.

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 which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

General Implementation Details

- Although you are not required to use more than one class per task, you are required to modularise classes properly. No method should be longer than 50 lines.
- Your coding style should be consistent with Java coding conventions (http://www.oracle.com/technetwork/java/codeconventions-150003.pdf)
- You should include comments to explain your code.
- It is not necessary to use dynamic data structures to store the input data (i.e. it is fine to define a fixed size data structure taking into account the maximum possible amount of input data).
- Your programs will be marked with Java SE 9.0. Make sure you test your programs with this setting before you make the submission.

Problem Description

This assignment is to continue the development of Assignment 1, MiniNet.

The scenario is still the same as that in Assignment 1. A social network is a collection of connections linking a set of people. There are three types of person: **Adult** (over 16), **Child** (3 - 16 year old) and **YoungChild** (2 year old or younger). Each person has a "profile" in the social network.

Your application still needs to keep track of persons' name associated with that profile, an optional image that the person may wish to display with his/her profile. In addition an optional "status" attribute is available for the profile. This is basically just a String indicating what activity the owner of that profile is currently engaged in, such as "working at KFC", "student at RMIT", "Freelance", "looking for jobs". Of course, each person has a list of connections on his/her profile.

A child can be a friend ONLY to another child who is also younger than 16 and from a different family even that between a "17-year-old adult" and a "16-year-old child". The age difference between these two young friends cannot be more than 3 years. A young child cannot have any friends but may have siblings.

Other than "friend" and "parent" relations, there are other types of connections between people. Your application should support Colleague and Classmate relations. They represent that the two people are working together and studying together respectively. Colleague relation only applies on Adults. Classmate relation can apply on Adults and Children, but not young children. [*You may consider interfaces in this scenario*.]

Note, we assume there is no ternary relation on in MiniNet. An adult may have no connections.

Important: Software requirement may change. Please pay attention to the course announcement.

Part 1: Design (3 marks)

Update the class diagram of your design. New classes such as the helper classes for handling exceptions, reading files and so on also need to be added to the diagrams.

You may use a design different to that in your Assignment 1 submission.

You will need to be able to answer the following questions with respect to your design:

1. Explain the changes if you use a different design compared to your assignment 1
2. Explain how the new classes are organized
3. Explain the process by which your program will interact with user and external data source to run a game.

Class Diagram Format is the same as specified in Assignment 1.

Part 2: Exceptions (3 marks)

Possible errors in your program should be handled by the Exception mechanism of Java. That include:

- **`TooYoungException`** when trying to make friend with a young child.

- **`NotToBeFriendsException`** when trying to make an adult and a child friend or connect two children with an age gap larger than 3.

- **`NoParentException`** when a child or young child has no parent or has only one parent, e.g. adding a child to one adult, or to two adults who are not a couple. That also happens when trying to delete an adult who has at least one dependent. (In this world a couple that have at least one kid become inseparable and immortal.)

- **`NoAvailableException`** when trying to make two adults a couple and at least one of them is already connected with another adult as a couple.

- **`NotToBeCoupledException`** when trying to make a couple when at least one member is not an adult.

- **`NoSuchAgeException`** when trying to enter a person whose age is negative or over 150. (You can treat age as integer)

- **`NotToBeColleaguesException`** when trying to connect a child in a colleague relation.

- **`NotToBeClassmatesException`** when trying to make a young child in a classmate relation.

Other types of exceptions that may occur during file opening, database connection, data management need to be handled properly.

Part 3: External Data Sources (6 marks)

Your program should be able to handle external data sources.

By default it reads an ASCII file "`people.txt`" which looks like this

```
Alex Smith, "", "student at RMIT", M, 21, WA
Ben Turner, "BenPhoto.jpg", "manager at Coles", M, 35, VIC
Hannah White, "Hannah.png", "student at PLC", F, 14, VIC
Zoe Foster, "", "Founder of ZFX", F, 28, VIC
Mark Turner, "Mark.jpeg", "", M, 2, VIC
```

You can assume that
1. the information is always correct.
2. age is an integer and is always correct.

3. possible states are `ACT, NSW, NT, QLD, SA, TAS, VIC` and `WA`.
3. no duplicate names. Hence the name alone can be used as the identifier.
4. profile image may be empty

Your program also read an ASCII file "`relations.txt`" which looks like the following:

```
Alex Smith, Ben Turner, friends
Ben Turner, Zoe Foster, couple
Ben Turner, Mark Turner, parent
Mark Turner, Zoe Foster, parent
……
……
```

You can assume that
1. there are no duplicates.
2. names in one relation are in alphabetic order.
   So, you see `Alex Smith, Ben Turner, friend`, but not `Ben Turner, Alex Smith, friend`.
3. a valid child record always has both parents listed in the record.
4. a record that violates any constraint will not be used. For example, `Hannah White` will not be added to the network as her parent info is missing. An error message should be displayed for such situation.
5. other relations are "`classmates`" and "`colleagues`"

When an **embedded** SQLite or HSQLDB database connection is found and there is no `people.txt` in the system, then your program should read the personal data from the database. Any updates on the personal data should be stored in the database as well.

Your program should show an error message on the user interface if it cannot find a database connection or "`people.txt`" on the local file system.

[1. Using databases for the relation information is not required in this assignment.]
[2. Updating the files is not required in this assignment.]

Part 4: GUI (10 marks)

Your program should have a graphical interface to facilitate user interaction. It should support the following functionalities through **mouse operations**:

- Add a person into the network
- Select a person
- Display the profile of the selected person (show image if available, and show no image text if empty)
- Delete the selected person. Note when one person is removed from the social network, all

his/her connections are affected, meaning the profiles of his/her friends, colleagues need to be updated as well. Also that person should not appear in the full list of all people.
- Find out whether two people are directly connected in some ways.
- Define relation between two people e.g. friends, parent, classmates
- Find out the name(s) of a person's child(ren) or the names of the parents
- Exit

Errors and exceptions need to be handled gracefully through the GUI as well, e.g. a missing file, deleting a parent, making two young children friends and so on.

You are free to design your own GUI as long as it supports the above functionality. The GUI should be user friendly and visually pleasant. You may choose different Java GUI packages. Using JSP and servlets is permitted but does not attract extra marks.

Your code must be runnable on RMIT lab machines and core teaching machines without installing additional packages.

Extension for Bonus (3 marks Bonus)

Finding friends of friends:

A key aspect of social networks is not only keeping track of how many people you have as friends, but also how you are connected with anyone on the network, e.g. through a common friend or through several degrees of connection.

You are to display the "connection chain" that shows the **shortest** relation pathway connecting one person to another. For example, if X is a friend of Y, Y is a friend of Z, and there is no relation between X & Z, then a relation chain exists between X and Z: X → Y → Z. Dealing longer and more complex chains can be a challenging problem.

Your program should show such chain between any two people that are selected by the user. "Not connected" should be displayed. If the two people are not connected, e.g. one is a completely isolated person.

Part 5: Proper Coding Practice (3 marks)

Your source code needs to follow the Java coding convention
http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Your program should enable proper tests e.g. with unit test embedded.

Your program is properly documented and can generate a reasonable HTML Javadoc.

Proper messages are shown on the command-line console to reflect frontend GUI operations.

Start-up

Your main startup class should be named **`MiniNet`**.

Submission Guidelines

You can complete this assignment in a group of two. No extra mark for individual submissions.

For group submissions, we expect about 50:50 split of coding between two group members. Each class should have the author name(s) stated in the first few lines of that class definition. It is a good idea that one class is written only by one person.

All submissions should have a **Github** repository, which records the progress of the development and so on. In the final submission, each group should nominate the contribution of each member, discuss possible issues and/or comment on your knowledge gain and your learning experience through this group work. Members of a same group may receive different marks.

Submission Details

Submissions should be made via Canvas as zip file of your project.

In addition you need to make a version (release) named **`assignment2`** in your Github repository. You can have a folder for your designs and documents in your GitHub as well.

You can submit your assignment as many times as you want before the due date. The last submission will overwrite the previous ones.

This assignment also has a lab demo component. Details will be posted on the course announcement.

Bonus mark percentage can be added for early submissions based on the mark you can achieve:

10% Bonus will be added if you submit 5 days earlier.
There is no bonus mark for submissions made one to four days earlier or marked below 50/100.
You can choose to submit early or complete the extension to claim the bonus, but NOT BOTH. Early submissions with the extension part implemented will be treated as on-time submissions.

Design documents should be in one of the following formats only and should be named as **design** (e.g. design.doc, design.pdf, design.gif, design.jpg or design.png)

- Microsoft Word format (must be compatible with Microsoft Word on RMIT CS terminals)
- Adobe PDF format
- GIF, PNG or JPEG image format (please do NOT submit BMP files)

Submission due date: by 11:59PM Tuesday 22 May 2018.

You should archive your design documents and code, and submit them as a "zip" file (no RAR or 7-Zip).

Please do NOT submit compiled files (*.*class* files), or you will get zero.

You will get zero if the submitted code cannot be compiled.

You may get zero if the submitted code is NOT the same as your version in Github.