

School of Computer Science & Software Engineering

Assignment 4 (Individual)

CSCI213 – JAVA PROGRAMMING AND APPLICATIONS
April 2016 – May 2016

UOW Moderator: Dr Lei Ye
Lecturer: Mr Aaron Yeo
Email: simcsci213@gmail.com
Weightage : 10% of Subject
Submission: 26th May 2016 2355hrs (Wollongong Time) to Moodle
Demo: 27th May 2016 during Lab 7

CSCI213 Assignment 4: Multithreading / Networking

1. Overview

This assignment has two main objectives. It aims to establish a basic familiarity with the Java API (Java Thread class) for creating a multithreaded Java application, and the Java networking packages for creating a client-server Java application.

2. Objectives

On completion of this assignment a student should be able to write Java applications that:

- Apply Java multithreading concepts
- Make use of Java API "Net" package
- Make use of Java API "Swing" and "AWT" packages
- Handle generated events
- Comprises classes designed based on object-oriented concepts
- Makes use of Java programming fundamentals

3. Scope

You are required to enhance the one player game application "Fishing Pair" done in Assignment 2 and 3 to a multi-players networking game (client-server architecture).

For this game application, the GUI should be run as a client, and all the computations for controlling the game should be carried out by a server. Your server should be able to serve multiple clients concurrently, meaning that it should be a **threaded server**. Besides providing the required functionalities, your program should incorporate appropriate error handling. Comments are also to be inserted to improve program clarity. Before you start coding your program, you are strongly advised to carry out proper problem analysis and program design. You are expected to explore object-oriented concepts to design appropriate classes for this

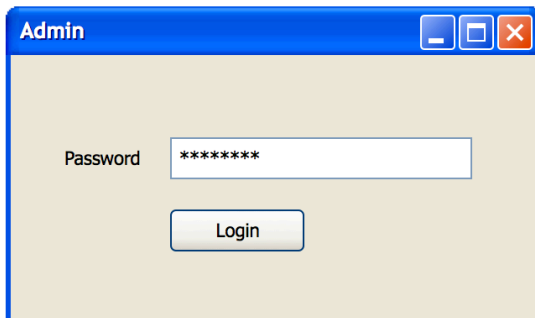
4. Requirements

There are two sub modules in this program; Client and Server.

4.1 Server Module

4.1.1 Admin Login

This dialogue box allows the admin to login to the system.



A Windows-style dialog box titled "Admin" with a blue title bar and standard minimize, maximize, and close buttons. The dialog has a light beige background. It contains a label "Password" followed by a text input field filled with asterisks "*****". Below the input field is a button labeled "Login".

4.1.2 Admin Main menu

Upon successful login, the administrator will be presented with the main menu.



A Windows-style dialog box titled "Game Server" with a blue title bar and standard minimize, maximize, and close buttons. The dialog has a light beige background. At the top, it says "Game Admin". Below this, there is a grey rectangular area containing a "Port" label and a text input field with "4444", and a "No. of Player(s):" label followed by a dropdown menu showing "1". Below the grey area is a button labeled "Start Game Server". At the bottom of the dialog, outside the grey area, is a button labeled "Logout".

The admin can specific the port number that the server listens to (default is 4444). Then the admin selects how many players (1 to 3 players) for the game.

After the admin clicks on "Start Game Server", the server will be started and listens for new connections.



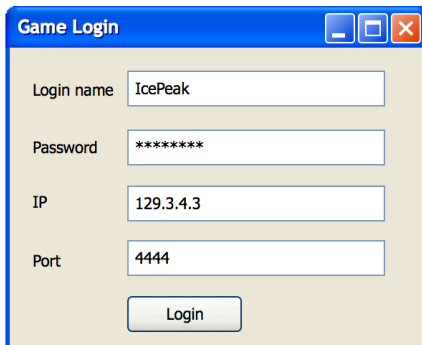
Textbox to update the status of each player. The server will continuously update the status of each player (e.g. Waiting, Connected, Wait for IcePeak, Quit)

(Assume the admin has selected 3 players for this game)
The server will stop by itself when all the player choose to exit the game.

4.2 Client Module

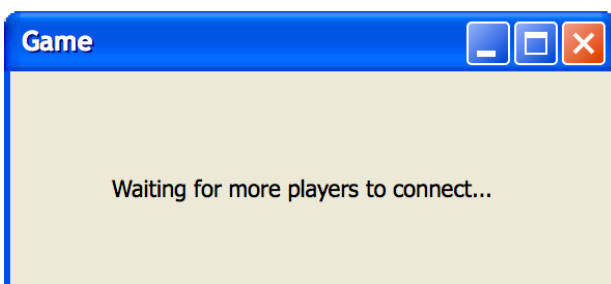
4.2.1 Login

The game starts by the player logging into the game.



4.2.2 Waiting

Upon logging in, if the server is still waiting for other player(s) to get connected, it will display a waiting message until all the players have connected to the server.



4.2.3 Play Game

The game starts after all the players are connected to the game server.

The GUI design is similar to those in Assignment 3.

You may implement this module without using GUI.

The status of each player will be updated at the server admin screen.

Each player can view the status of other player's status (e.g. Waiting, IcePeak select Card 3 and 5, Exit) on their screen as well.

4.2 Network related Error Handling

Your program should be able to handle **network type** of error situations, for examples:

- where a player enter a wrong server port number and is unable to connect to the server.
- the server timeout when one of the player did not connect to the server after certain amount of time.

4.4 Network related Enhancement

You are free to do any Network related enhancements to the game and write those enhancement in the report.

Possible network related enhancement examples:

- Encrypt all the commands sent from client to server and decrypt at the server.
- Support different types of encryption standard.
- Allow admin to select different type of encryption standard.
- Allow admin to set the timeout period.
- The players are able to "private message" each other.

5. Submission

A complete submission requires the following items:

1. Report
2. Program in Zip file for execution
3. Program listing in a single word file (for turnitin check)

Missing in any of the above items is consider non-submission.

Late submission for any above items is consider late submission.

Program file header

For each Java program file, provide the header as shown

```
/*
 * CSCI213 Assignment 4
 * -----
 * File name: (state name of .java file)
 * Author: (State student name in FULL)
 * Student Number: (State UOW student number)
 * Description: (A brief description of this class)
 */
```

Any late submission of work must be accompanied by an application for Special Consideration, requested via SOLS. Unless an extension is granted, any late submission will receive a penalty of 25% of its total worth per day including weekends, and will result in zero mark being recorded on or after the 4th late day. Request for extension with supporting document must be submitted to SIM administration for further consideration before the submission date and the tutor must be informed. Extension will be granted on a case-by-case basis.

Report requirements

The report to be submitted consists of the following sections:

1. Cover page – clearly state your name and UOW student's number.
2. Content Page
3. Classes diagram
4. Test-run of Program: You have to provide screen outputs to show the correct execution of your program according to the requirements stated.
5. Enhancements: List down and explain the enhancements that you have done. Provide screen captures here
6. Conclusion and Reflection

Testing requirements

Make sure that you are able to run your program using the command (**java FPGServer** and **java FPGClient**) directly from the command prompt from the project folder.

In the players.dat, you should have the following FOUR players' data.

Login Name	Password	Last login date	Scores
player1	p1	2016-1-11	10
player2	p2	2015-12-2	12
player3	p3	2016-2-7	15
player4	p4	2016-3-15	9

Refer to Report guideline and marks allocation document for detailed report requirements.

6. Plagiarism

The University's policy on copying does not allow you to copy software as well as your assessment solutions from another person. Copying of another person's work is unacceptable. It is the responsibility of all students that their assessment solutions are their own work. You must also ensure that others do not obtain access to your solutions for the purpose of copying a part of them.

Where such plagiarism is detected, both of the assessments involved will receive ZERO mark.

7. Evaluation Criteria (total 50 marks)

Refer to Report guideline and marks allocation document