# SEEM3460/ESTR3504 (2014)

# Project

*Due on December 8 noon (11:59AM), 2014*

## General Information

- ✧ 30% or more mark penalty for uninformed late submission.
- ✧ You must follow the guideline in this file, or there will be a mark penalty.
- ✧ You are advised to develop your program under a Java IDE (e.g. NetBeans).
- ✧ All development and running of the Java programs in this project are **all in Windows** platform.

## Faculty Guideline for Plagiarism

If a student is found plagiarizing, a very heavy punishment will be imposed. The definition of plagiarism includes copying of the whole or parts of programming assignments from other's work or the Web. The penalty will apply to both the one who copies the work and the one whose work is being copied.

## Problem Overview

The objective of this assignment is to complete the game "Higher-Or-Lower" in Java by implementing several methods. The project also has an **EXTENSION PART** for to let you extend the program in your own way. Details are given in the Problem Specification section below.

You must finish the task **ONLY** using the **CODE** files **HighLow.java** and **GUI.java** since all other files will not be marked. See the Submission section for details about the submission procedure.

Material for this project can be found in this folder:

~seem3460/distribute/proj1
or http://www.se.cuhk.edu.hk/~seem3460/assignment/HighLow_demo.rar

A finished program example (**WITHOUT EXTENSION**) is prepared to show you the game logic:

1) Download from http://www.se.cuhk.edu.hk/~seem3460/assignment/HighLow.rar to your Windows platform.
2) Unzip the file **HighLow.rar**
3) Double click **HighLow.jar** (Require jre or jdk 1.7.0 above installed).

# Problem Specification

## The "Higher-Or-Lower" Game

The game is played by 2 human players. At the beginning of a new game, one card is dealt to each player. The game has 9 rounds. In each round, an extra card will be dealt to each player. Before the card is dealt, the player guesses whether the coming card is higher or lower than his own last card dealt by clicking the "**Higher**" or "**Lower**" button. A correct guess wins 20 points while a wrong guess loses 10 points. All players start from zero points. After the rounds, the player who scores most points wins.

The "**New Game**" button, whenever it is clicked, should reset everything and start a new game. The "**Quit Game**" button should close the game window.

## Program Structure

An OOP structure is provided to you as the starting point. It contains 3 components: **HighLow_demo.java**, **GUI_demo.java** and **CardLayout.java**. Be reminded that you should rename **HighLow_demo.java** and **GUI_demo.java** to **HighLow.java** and **GUI.java** respectively for your final program submission. The main function is in the **HighLow** class. DO NOT modify anything in **CardLayout.java** since it is not part of your submission.

Here is a summary of the structure:

| File | Class | Functionality |
|---|---|---|
| **CardLayout.java** | CardLayout | Layout only used in GUI.Hand for packing cards |
| **HighLow.java** | HighLow | The class representing a game<br>HighLow **start**()<br>- return a new "Higher-Or-Lower" game<br>void **init**() set the action listener of buttons<br>void **cardShuffle**() shuffle the cardDeck<br>int **nextCard**() take a card from the cardDeck<br>void **actionPerformed**(ActionEvent e)<br>- assign methods to handle events (button clicks)<br>void **newGame**() initialize a new game<br>void **makeGuess**(int guess)<br>- handle a player guess and proceed the game<br>void **quitGame**() exit the game environment<br>void **endGame**()<br>- lock meaningless buttons and tell the winner<br>void **main**()<br>- the main function which starts the execution |
| | Player | Encapsulation of player information<br>void **reset**() clear hand cards and set score to zero<br>void **updateScore**(int i) update the score |

| File | Class | Functionality |
|---|---|---|
| | main() | Starting point of the program |
| **GUI.java** | CardIMG | Label for displaying picture of a card<br>int **toFileID**(int id) converts the rank ID to file ID<br>About card ID:<br>In most places of the program, cards are indexed using their rank:<br>0=♣2 < ♦2 < ♥2 < ♠2 < ♣3 < … < ♠A=51<br>But the PNG files in picture folder uses another index. So we need a function for conversion |
| | Hand | A GUI set displaying the player card holding<br>void **distribute**(int id) deals a card<br>void **reset**() remove all cards |
| | Scoreboard | A GUI set displaying the player name and score |

## The Task - Main Part

The following are the main tasks:

1. Complete the card ID mapping: **toFileID**(int id)
2. Complete **newGame**(), **makeGuess**(int guess), **endGame**() to make the game run smoothly

The MAIN PART constitutes 90% of the total mark. You are strongly advised not to modify any part except parts labelled with the "CODE HERE" comments. But the EXTENSION PART in general requires extra modification.

## The Task - Extension Part

The extension part is an open-ended task that constitutes 10% of the total mark. You may do any kind of upgrade to the program given that your final program keeps the meaning of the game well (otherwise marks will be deducted for your **MAIN PART**). Marks will be awarded according to the difficulty of the extension you made. You should also write a report not exceeding one page (details are given below).The awarded marks will be up to the **SUM OF VALUE** of simultaneous extensions you made, **bounded by 10%**. Some possible extensions are listed below.

Extensions with a value of 2%:

- To implement a good object-orient design, split HighLow.java into two files: HighLow.java and Player.java. The HighLow.java should contain HighLow class. The Player.java should contain Player class. Split GUI.java into three files similarly.

Extensions with a value of 5%:

- Make a graphical highlight to show either: the player in turn; or the score status
- Add a "Hint" button for showing the count of remaining higher/lower cards

Extensions with a value of 5%:

- Add a "Cheat" button for displaying the next card in the game interface

- Add a “Pass” button for not making a guess and gets zero point in a round
- Implement an AI player that plays South with a best/randomized strategy

Extensions with a value of 10%:

- Change the game design (including the interface) to make it a three-player game
- Write a class **GameFrame** in **GUI** so that **HighLow.java** no longer needs imports

**Note:**

If you complete the extension part, you should write a report not exceeding one page. The file name should be <student-ID>-extension-work.doc. The report should describe what you have done in the extension part and the meaning of each new class. If you replace the main function, you should also clarify the position of the main function in the report.

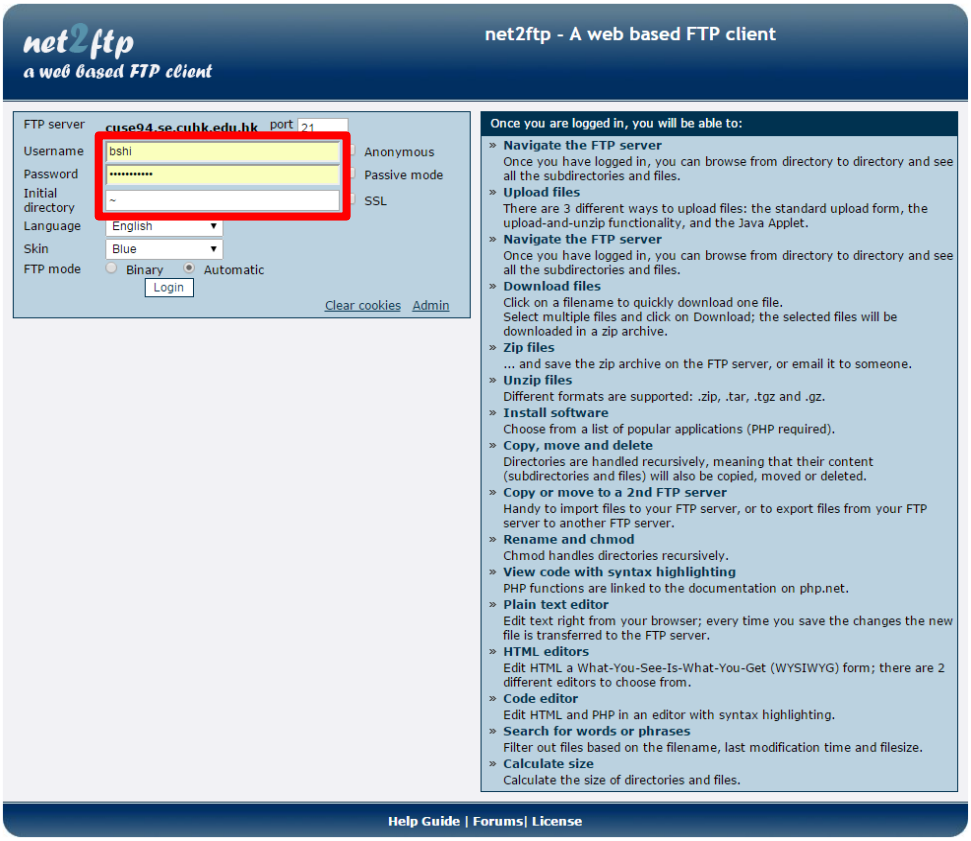
# Additional Requirements for ESTR3540

The main part and the extension part is the same for ESTR3540. However, the main part constitutes 80% of the total mark and the extension part constitutes 20% of the total mark.

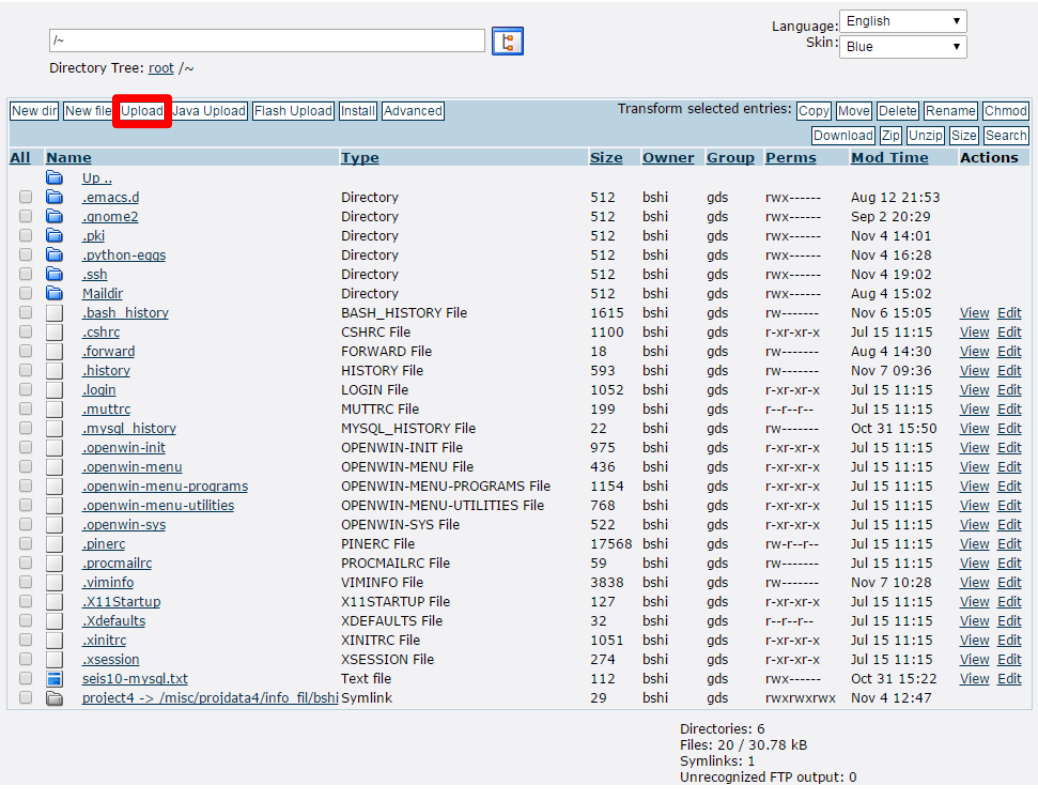
# Submission

Please follow the submission procedures, so that we could ensure your assignment is received properly.

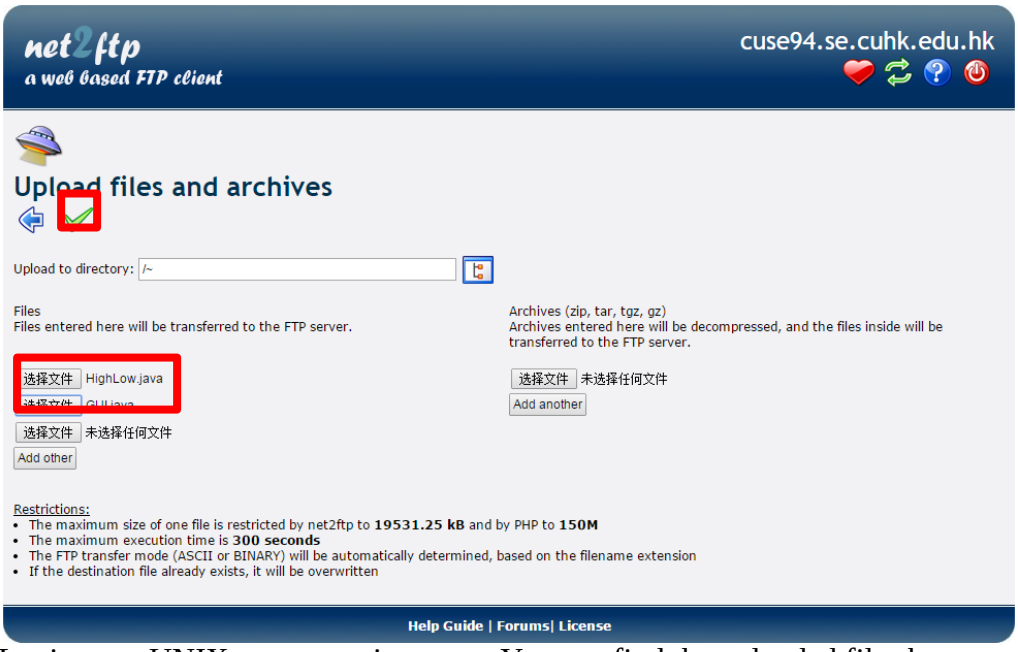
- ✧ Only submit your assignment **USING YOUR OWN** UNIX account!
- ✧ Transfer your source files and report (if exists) from Windows to UNIX using webftp service as follows:
  - ■ Open your browser and go to http:/webftp.se.cuhk.edu.hk.
  - ■ Login with your UNIX account.



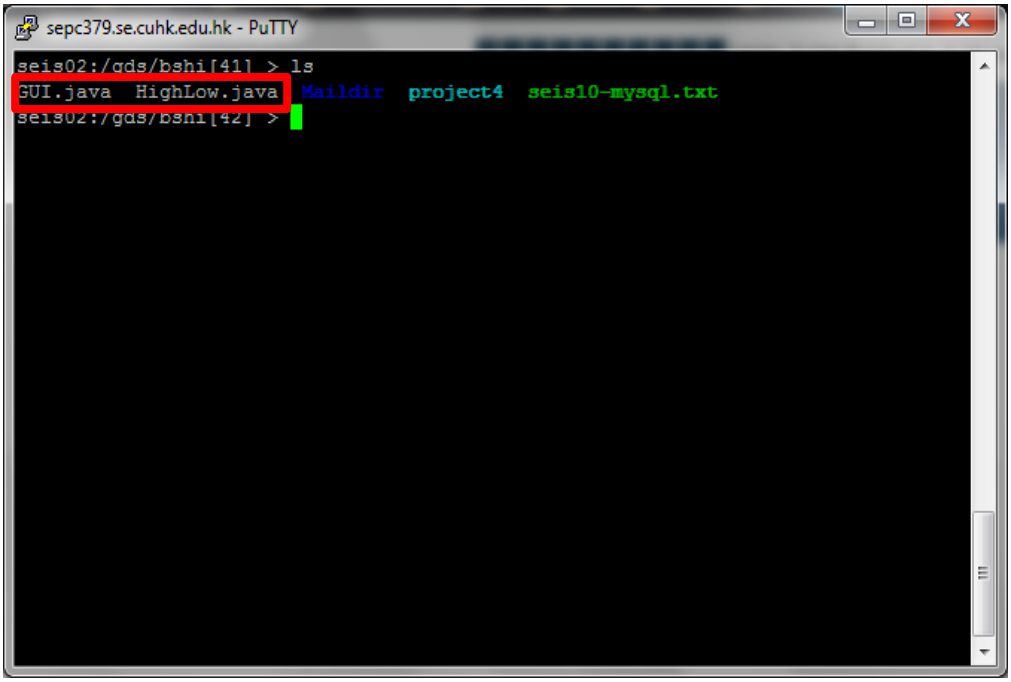

- Click the upload button




- Select the files that you will submit and click OK.




- Login your UNIX account using putty. You can find the uploaded files by command "ls".




- ✧ Compress your source files and report file (if exists) into a single archive as follows:
  **zip highlow HighLow.java GUI.java**
- ✧ Run the submit program specifying the assignment identifier and filename
  **~seem3460/submit proj1 highlow.zip**
- ✧ When the program asks your student ID, type in your **FULL** student ID as follows:
  **What is your full student ID? 1155001122**
- ✧ Then the program will summarize your personal information and ask for your confirmation. **FULLY** type in "**yes**" to confirm your information. If you typed in your

information wrongly, type “no” to exit the program and start again.

**Your student ID: 1155001122; Your account name: lwai**

**Is the above information correct? (Y/N) Y**

- ✧ Then you should see a message like this:
  **Connecting to SEEM3460 Submission Server...**

  **sftp channel opened and connected.**

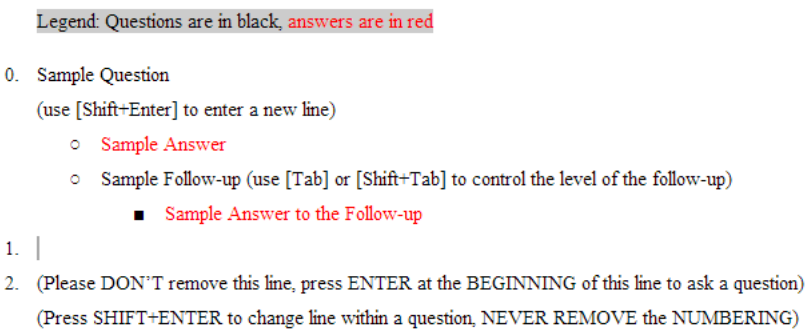
  **Uploading...**

  **Done! Thanks.**
- ✧ If you see any additional error message apart from the above, then your assignment may not be submitted properly. Try to submit again. If the problem persists, log down the error message and send an email to **bshi@se.cuhk.edu.hk** to clarify, and attach your file to submit in the email. Improper use of email submission will cause mark deduction.
- ✧ Multiple submissions are not encouraged, but allowed when necessary. Only the last submission will be graded.

# Question & Answer

With any questions about the assignment, please check the following Google Document for Q&A first and write only new questions that have not been asked yet:

https://docs.google.com/document/d/1fZU3ss5Vc6emOXM_HMSWCGGj1B-0NRTRuzoA1fpuzFo



Legend: Questions are in black, answers are in red

0. Sample Question
   (use [Shift+Enter] to enter a new line)
   - Sample Answer
   - Sample Follow-up (use [Tab] or [Shift+Tab] to control the level of the follow-up)
     - Sample Answer to the Follow-up
1. 
2. (Please DON’T remove this line, press ENTER at the BEGINNING of this line to ask a question)
   (Press SHIFT+ENTER to change line within a question, NEVER REMOVE the NUMBERING)