

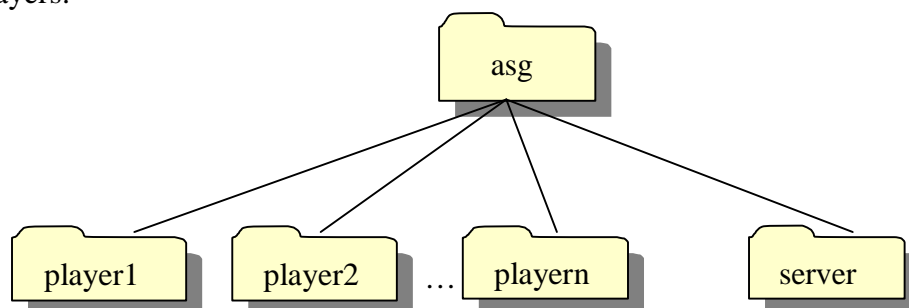
## COMPSCI334 Assignment 1

The objective of this assignment is to practice the use of Java RMI technology. The system to be developed consists of one server and several players. When the system is deployed, the components are expected to be located at different sites.

The system is a simplified networked Tic Tac Toe game. A Tic Tac Toe game is to be played between two players. Only registered users can use the system to play the game. The server keeps information about the registered users. The information should include UPI and password of each user. The UPI should be used to identify a user uniquely. The server should also record the number of games that each user has won, lost and drawn.

When a player wants to play the game, the player must first log on the server. When a player logs in, the server provides the player with a list of players who are currently available to play a game. A player is available to play a game if the player has logged on the server and is not playing a game with another player. A player can choose to play a game with a player in the available players list. When two players agree to play with each other, the two players interact with each other to play the game directly without the intervention of the server. When two players finish playing a game, they should be put back to the available players list and the server should update the information regarding the number of wins, loses and draws about the two players according to the result of the game. The interactions between a player and the server and the interaction between any two players should be carried out using Java RMI.

Your assignment folder should have the following structure. The playerx folder holds the programs for player x. The server folder holds the server programs as well as the database which stores the players' information. For this assignment, you should have five players.



When the system is deployed, each folder corresponds to a machine (i.e. a site). The machines are connected by a network.

To make the marking of your assignment easier, you should NOT require the marker to upload class files (if any) to a web server before running your programs.

### Part 1 (10 points)

On the server site, there is a database and programs for interacting with the database. The database contains a table 'users' which contains the registered users' information. The table can be created with the SQL statement below.

```
CREATE TABLE users (UPI char (7) NOT NULL CONSTRAINT tableConstraint PRIMARY KEY, password char (8) NULL, win int NULL, lose int NULL, draw int NULL)
```

UPI is the UPI of a user. The meaning of the other columns should be obvious.

Write a program called DBinit.java. The program should initialise the database on the server. It should be assumed that the database is an MS Access database which is

named as users.mdb. The file holding the initial data for populating the database is a text files which is named as USERS.CSV. The text file consists of several rows. Columns in a row are separated by “,” as shown below. The columns correspond to the columns in the users table.

```

player1,player1p,0,0,0
player2,player2p,0,0,0
player3,player3p,0,0,0
player4,player4p,0,0,0
player5,player5p,0,0,0

```

Note: Your program should initialise the databases correctly regardless whether the databases are empty or not. In other words, DBinit can be executed multiple times.

## Part 2 (40 points)

Write a program which allows a player log on the server. When a player logs in successfully, the server should provide a list of available players to the player. The interface for logging in and displaying available players list are shown below.

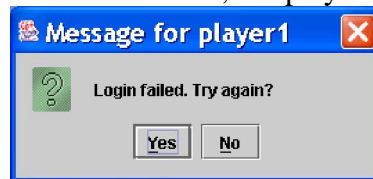
### Player Login Interface

When a player starts his/her program, the following should be displayed:



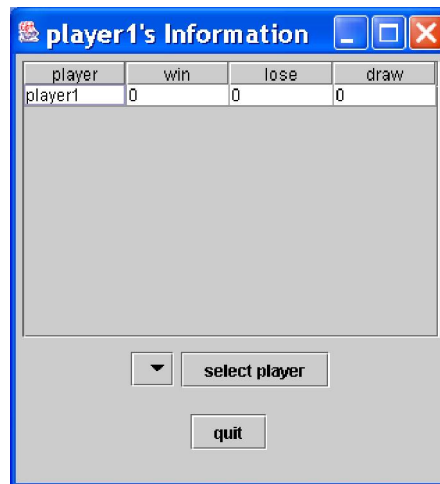
Figure 1

When the “login” button is clicked, the information entered by the player should be sent to the server. A login attempt is successful if the UPI and the password matches the corresponding information stored in a row in the database on the server. If a login fails, the dialog panel below should be shown. The title of the panel must include the UPI of the player. If the “Yes” button is clicked, the login frame in Figure-1 should be displayed again. If the “No” button is clicked, the player’s program terminates.



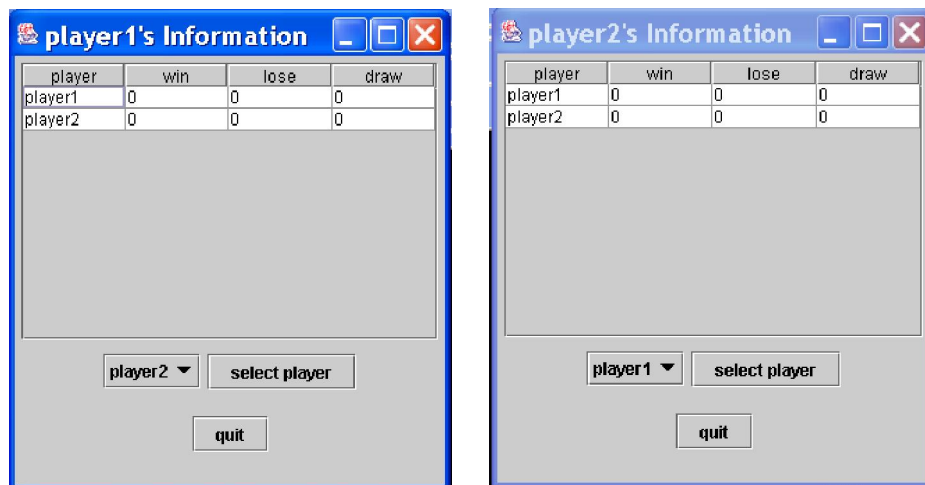
### Player Information Frame

If a login is successful, a *player information frame* as below should be displayed:



The title of the frame must include the UPI of the player. At the top of the frame, it is a table which shows the information about the players who are currently available. In this case, only one player, *player1*, has logged in. Therefore, only *player1* appears in the table. The combo box and the “*select player*” button will be explained in Part 3. If the “*quit*” button is clicked, the player’s program terminates.

Assume that another player, *player2*, has logged in successfully. After *player2* logged in successfully, the following frames should be seen by *player1* and *player2* respectively.



It should be noted that the table showing the available players in *player1*’s frame has been automatically updated.

### Part 3 (20 points)

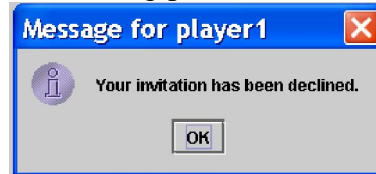
You are required to improve the program in Part 2 to allow a player to select a player in the available players list to play the tic tac toe game.

The combo box in a player’s information frame should include all the available players apart from the player himself/herself. A player uses the combo box to select a player. In this case, assume that *player1* selects *player2* to play. When the “*select player*” button is clicked, the following dialog panel should be shown on the selected player’s monitor.

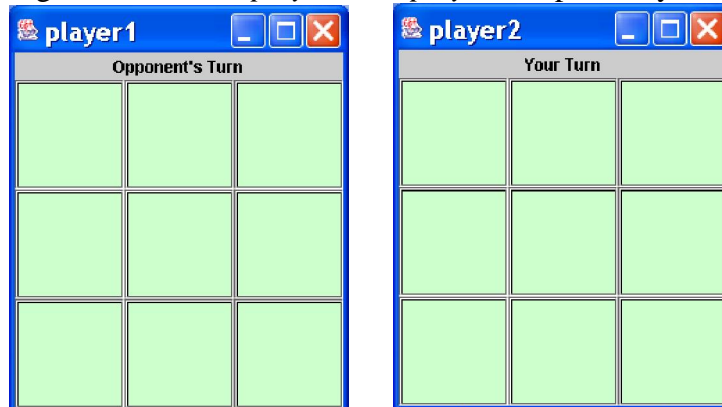


Note: The UPI of the player should appear in the title of the dialog panel.

If *player2* clicks “No” button, the dialog panel below should be shown to *player1*.



If *player2* clicks “Yes” button, a game board should be shown to each of the players and the players’ information frames should disappear. The player who initiates the game (i.e. *player1* in this case) will use white stone and the player who accepts the invitation (i.e. *player2* in this case) will use black stone and move first. The following figure shows the game boards for *player1* and *player2* respectively.

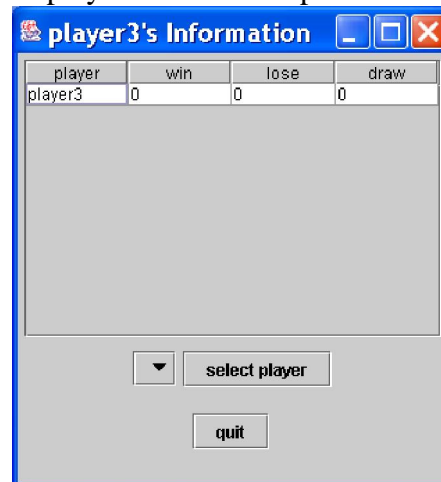


Note: The title of the board should display the UPI of the player. At the top of the board, you should indicate whose turn it is now.

When two players agree to play a game, they should be removed from the available players list. For example, assume that a third player, *player3*, logs in before *player1* and *player2* decide to play a game together. After logging in, *player3*’s information frame should be as below:



After *player1* and *player2* agree to play a game together, *player3*'s information frame should be as below. It should be noted that the available players list and the combo box displaying the available players have been updated automatically.



player	win	lose	draw
player3	0	0	0

Note:

- There are many possible scenarios regarding the timing of players inviting each other to play a game. It is your job to consider all these scenarios and provide a solution that works.
- In this part, you are not required to implement the code for playing the tic tac toe game.

#### Part 4 (30 points)

In this part, you are required to write the code for playing the tic tac toe game. When a game is completed, the server should update the information about the two players according to the result of the game.

Assume that (a) three players, *player1*, *player2* and *player3*, have logged on the server, (b) *player1* and *player2* have agreed to play a game, and (c) *player1* uses white stone and *player2* uses black stone. The game boards seen by *player1* and *player2* are as below:

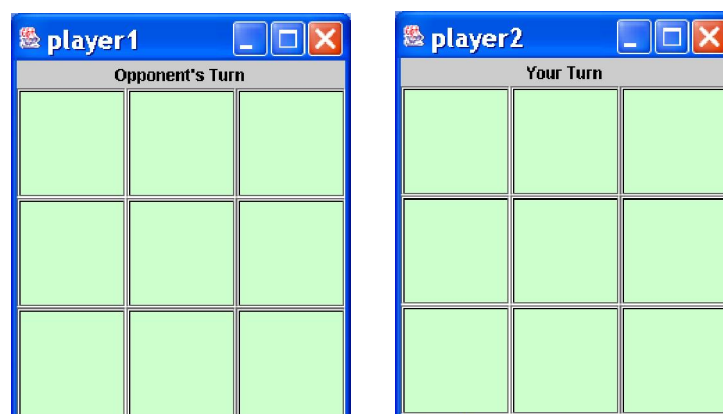


Figure 2

Your program should make sure that

- A player can only place a stone on the game boards if it is the player's turn. For example, in Figure-2, since it is *player2*'s turn, nothing happens if *player1* clicks on his/her game board.

- When a player places a stone of his/her colour on the board, his/her opponent's board should also be updated accordingly. Figure-3 shows the boards seen by *player1* and *player2* after *player2* made his/her move. It should be noted that the message regarding whose turn it is now is also updated accordingly.



Figure 3

- A player can only place a stone of his/her colour in a square which is still empty on the boards. In Figure-3, if *player1* clicks the black stone at the top-left corner, nothing happens.
- When a game is finished, a dialog panel should be shown for each of the players to inform them the result of the game. Figure-4 shows the dialog panels shown to *player1* and *player2* if the game is won by *player1*. Figure-5 shows the dialog panels shown to *player1* and *player2* if the game is a draw.



Figure 4

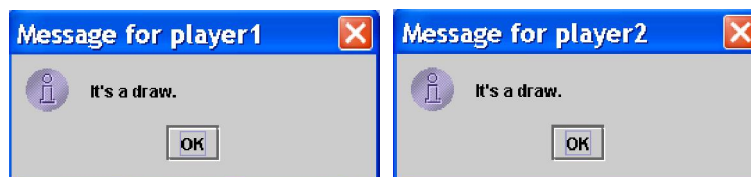


Figure 5

When a game is completed, the two players who played the game should be added to available players list again and they should be shown their information frames. Assume that *player1* won the game. Figure-6 shows what the three players should see after the game. It should be noted that *player1*'s and *player2*'s information have been updated.



Figure 6

You should test your program for at least five players logging on the server at the same time and at least two pairs of players playing games simultaneously.

### Report

You should write a report which covers the following issues:

- The parts that you have completed.
- Detailed instructions on running your program. You should include a batch file for running each of the components. [Note: It is important to provide as much details as possible. If your program cannot be executed by a marker, you will NOT get any mark.]

### Resources

You can download [Asg1Resource.zip](http://www.cs.auckland.ac.nz/compsci334s1t/assignments/Asg1Resource.zip) at “<http://www.cs.auckland.ac.nz/compsci334s1t/assignments/Asg1Resource.zip>”. The “example” folder in the file contains a program that provides a GUI interface to our Account example discussed in our lectures. You might find that this example is helpful for you to design the GUI interface for this assignment. The “TicTacToe” folder contains a program that implements a TicTacToe board that might be helpful for you to do this assignment. The “server” folder in the “Asg1” folder contains a database and a CSV files.

**Submission**

You **MUST** thoroughly test your programs in Tamaki lab before you submit your assignment. You should pack your programs into file Asg1.zip using a program like WinZip. You should make sure that the directory structure is preserved. The file which contains your report should also be included in Asg1.zip. Name your report file as "Report.doc". You should submit file Asg1.zip through the assignment drop box.

**Total Points: 100 Points**

**Due Date: 23:59, 2<sup>nd</sup> April 2008, sharp.**

**Electronic submission through the assignment drop box only.**

**No email submission will be accepted.**

**Appendix Tic-Tac-Toe Rules**

For a player, the objective of Tic Tac Toe is to get three stones of the player in a row (the three stones can be in a vertical, horizontal, or, diagonal line). If none of the players manage to achieve the objective, the game is a draw.