

Parallel & Distributed Processing

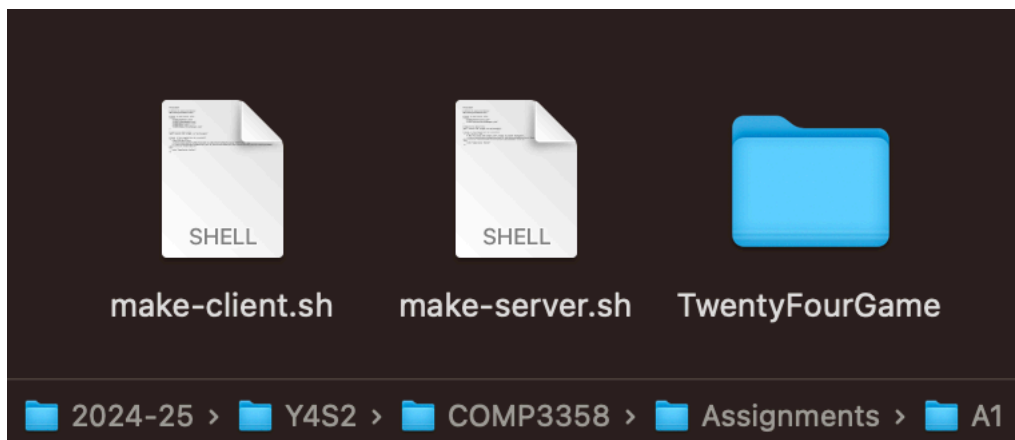
Java RMI — Setting up Login/Registration

The provided Java application allows for user login/registration, saving user records to a file — this is to be replaced with a database later.

Setup & Execution

The application is bundled as a package titled *Twenty Four Game*. For the convenience of the user/tester “build scripts” have also been provided. The package has been thoroughly tested on both MacOS (x86) & Ubuntu. To build and execute the program, follow the proceeding steps:

1. Change your directory to the one which contains the package along with the build scripts [see figure below].



2. Ensure that your system has Java 8 (also know as Java 1.8). Then run the following commands, in order:

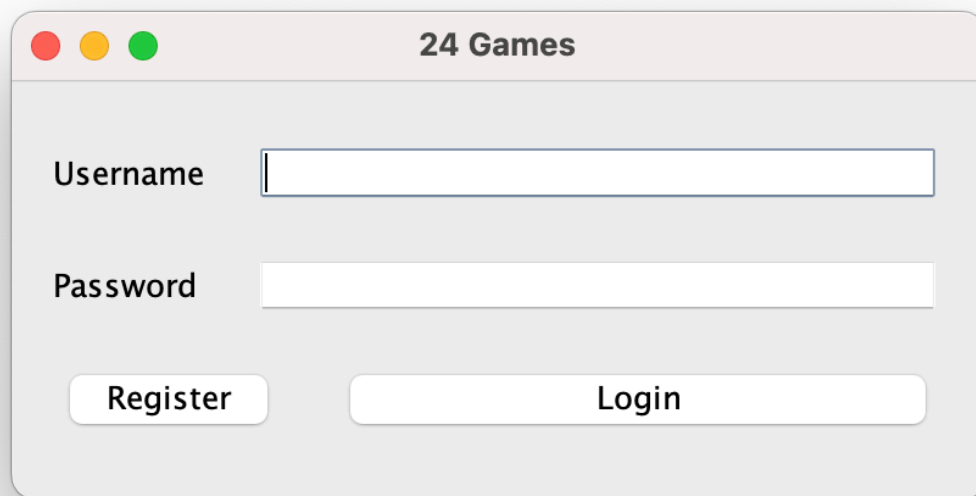
```
rmiregistry &  
bash make-server.sh &  
bash make-client.sh localhost
```

The first command sets up the rmi-registry locally, while the second command sets up the server locally where the rmi registry is launched. This can be set up remotely as well. Then for the client program, we would be required to pass in the ip of the remote server. For convenience we are testing everything locally. If the user/tester has access to multiple terminal sessions then running the first two commands in a separate shell than the last command will be *cleaner*.

Upon successfully executing the first two commands you should see the message `AuthenticationManager service registered`, along with other warnings about deprecation and security that can be ignored safely because our application has no users to steal money or data from.

If you are using a different version of Java you can modify the script to point to the specific version of Java 8 by modifying the script (e.g. replacing `Java` with `./path/to/my/Java_8/binary` in both the *make-client.sh* and *make-server.sh* scripts.

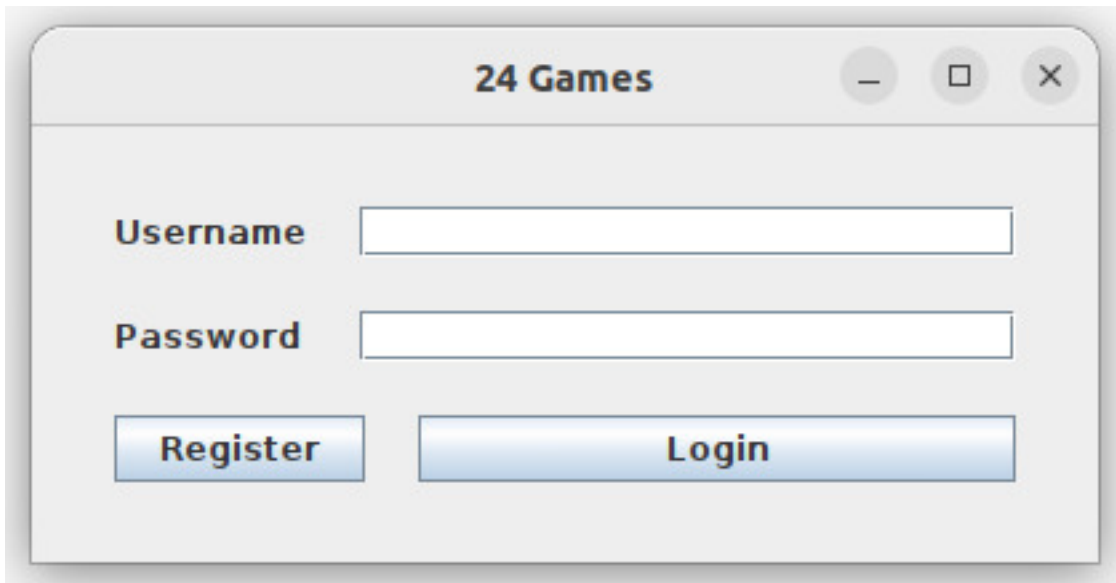
3. Once these commands have run, a window running the application should pop up as long as your system has a GUI. If not please consider switch a system that has once to use the application.



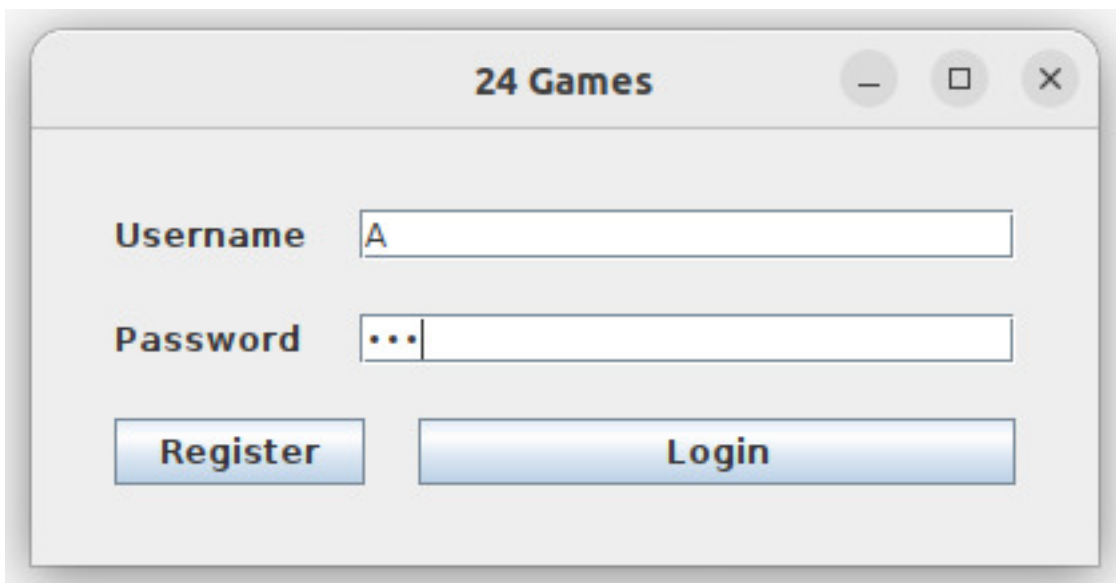
Once tired, you may want to run the command `rm TwentyFourGame/*/*.class` to clean the package.

Features

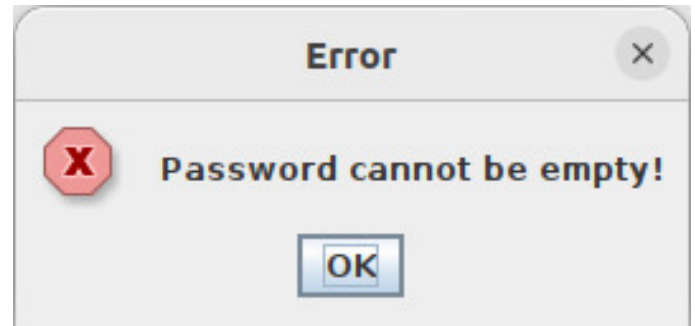
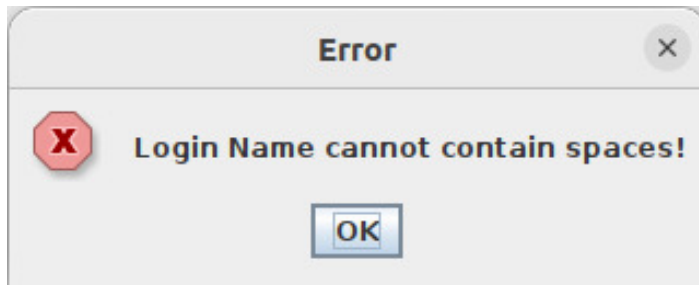
The remaining part of the report demonstrates the instance of the application running on an Ubuntu machine (as opposed to Mac OS on which the earlier instance of the application is running), highlighting the features.



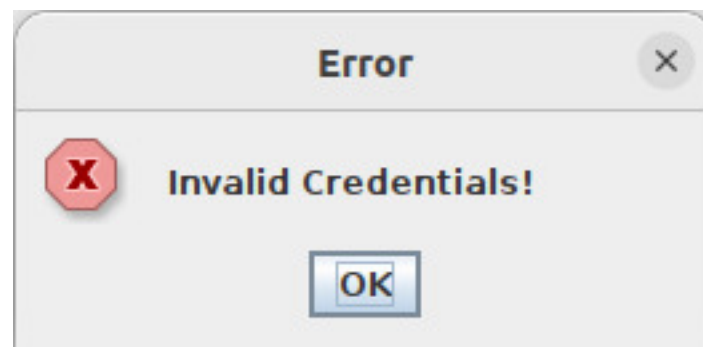
Upon entering the system, the user is greeted with the following window,



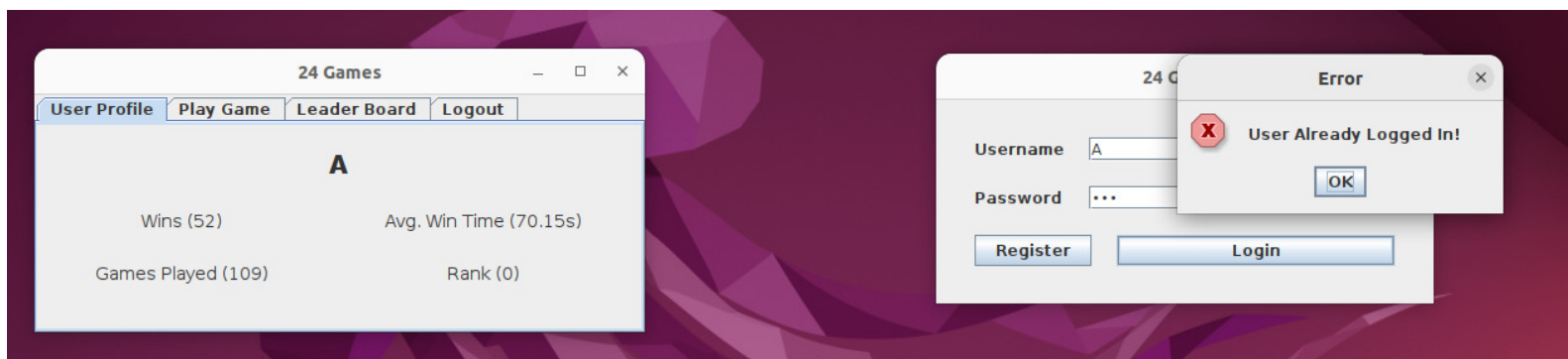
In case invalid login details are passed such as login names with spaces or empty passwords then we see the following prompts:



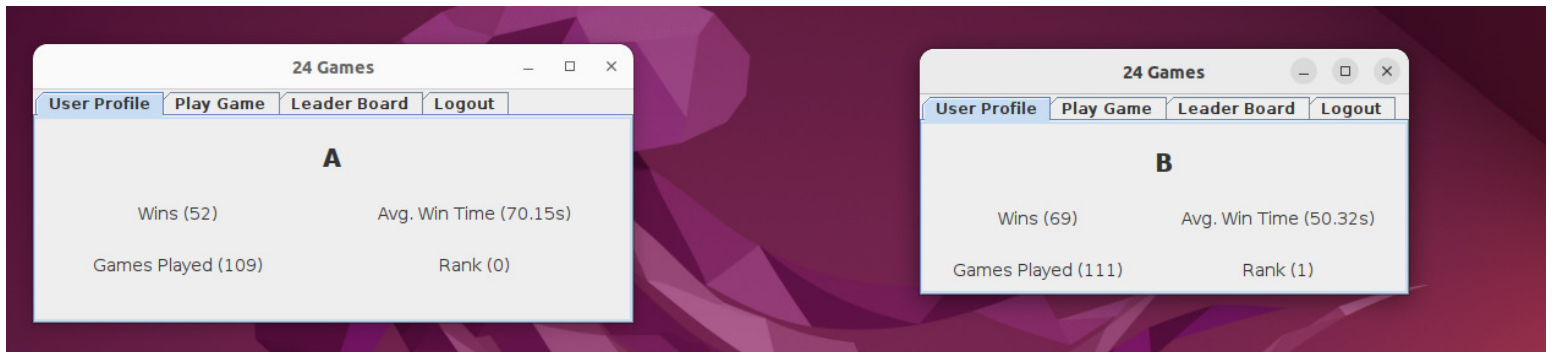
In case a non-empty password is provided alongside a legal username, then after a call to the server if the credentials (either username/password) are incorrect we get the following prompt:



Additionally, if another instance of the client application is run and that instance attempts to login to the same account as an already active user then their request will be rejected as can be seen in the following figure,



However, users can login to different accounts simultaneously without issue as seen here:



Upon login, the user is greeted with the following page where they can navigate through the tabbed panes,

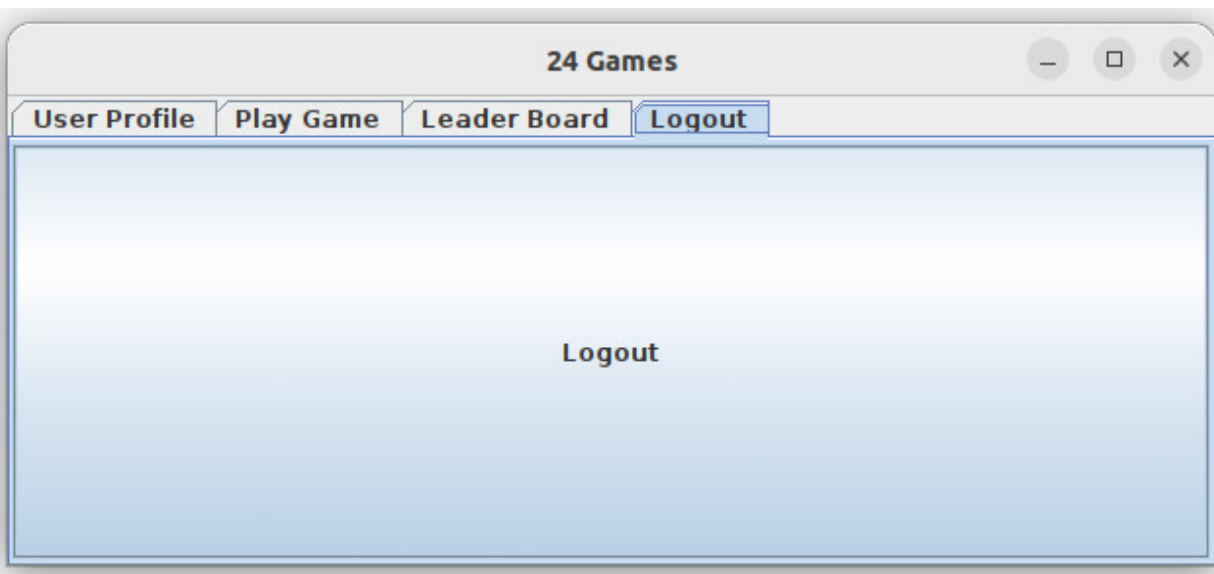


The game panel has been left empty as no requirement for its design was imposed.

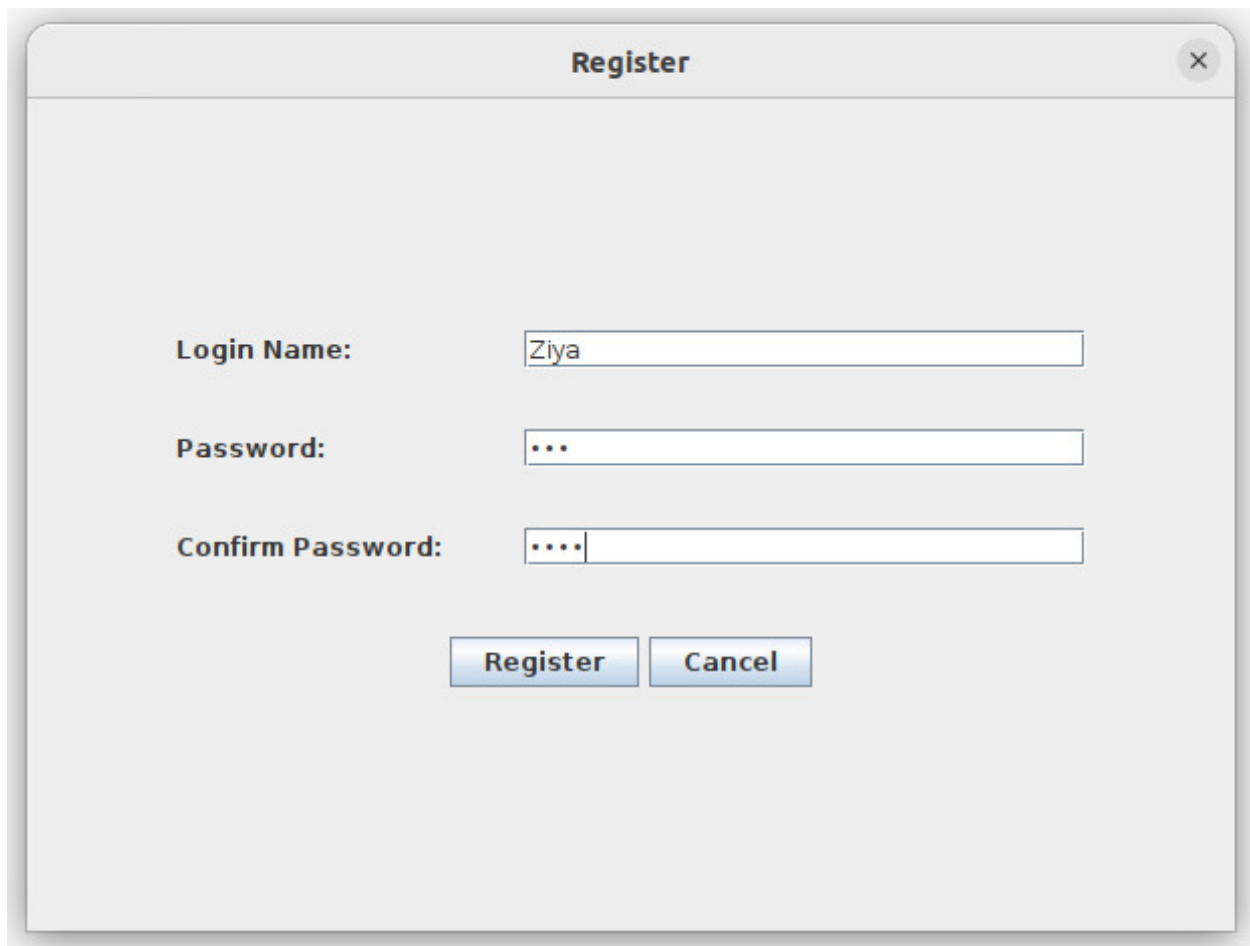


The screenshot shows the same "24 Games" window, but now the "Leader Board" tab is selected. The main content area displays a table with the following data:

Rank	Player	Games won	Games played	Avg. winning time
A	1	52	109	70.15
C	3	81	173	23.52
D	4	7	88	83.04
ABC	5	95	119	27.72
B	2	69	111	50.32

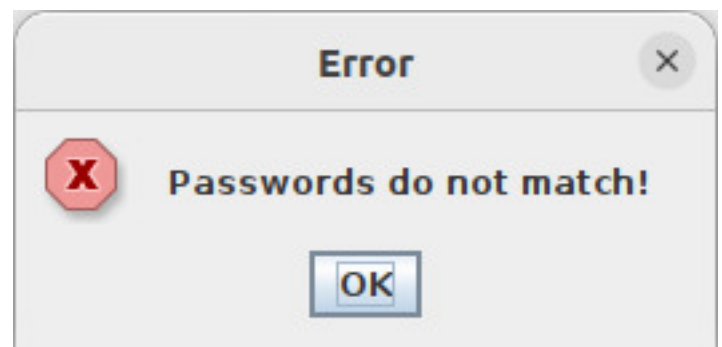
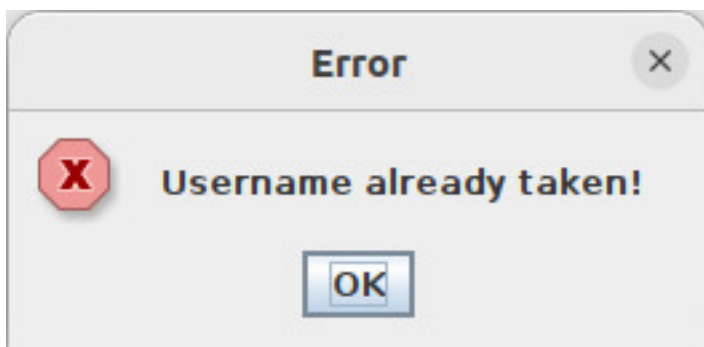


The user can also choose to register themselves by selecting the register button, in which case they will be greeted with the register panel.



A screenshot of a 'Register' dialog box. The dialog has a title bar with the word 'Register' and a close button (X). Inside, there are three input fields: 'Login Name:' with the text 'Ziya', 'Password:' with three dots, and 'Confirm Password:' with four dots. Below the fields are two buttons: 'Register' and 'Cancel'.

If the passwords entered don't match, or the username is already taken we receive a response from the server saying thus.



If you investigate deeper into the code base itself you will see that some security design decisions have been made. For example, instead of directly storing the passwords a hash (SHA-256) is stored instead.

There are even more error prompts that have been omitted from this report for the sake of brevity (such as for server disconnection etc.).

The primary file of interest in the server sub-package is the ``AuthenticationManager.java`` file, which contains the business logic for the server.

Whereas, in the client sub-package, the ``AppPanel.java`` file contains the UI for the user after being logged in and ``LoginManager.java`` and ``RegistrationManager.java`` as the names imply handle login and registration logic.

All other files are either utility modules, define interfaces or enums.

ChatGPT was used to aid design the UI for this project but the code was stringently reviewed by the developer and all important logic, including the read-write lock in the server, was written by the developer— me :) —as well!