This distributed implementation of a game of hangman allows several players to collaborate across multiple games whilst program logic is handled by a separate server application.

Potential concurrency issues in situations where multiple users are contributing to a single game are solved via use of the synchronization keyword to lock on key shared resources.

The general structure of the program follows Tanenbaum's model for Client-Server architectures, minus the data level, as no data is persisted other than a dictionary for words and a log of actions performed on the server.

The solution is split into 3 main projects.

- Server (HangmanDealer) - Handles connections to users as well as setting up multiple hangman games and . Gracefully handles user disconnects and other communication errors. Uses TCP over UDP to guarantee higher reliability. Also included in the files is a list of potential words included as a dictionary-style text file and a log of every completed game
- Basic Client (HangmanPlayer) - A basic client that allows a user to connect, choose/create a game room and play a single game with input/output provided by the command line.
- Web Client (HangmanWebFrontend) - Version of the basic client with basic HTML GUI that sends and receives input via AJAX interactions with a Java EE servlet. Each user is assumed to have a servlet, as such multiple games on one machine must each have their own instance of glassfish/etc. running

Also included in the Server class is an example of a set of unit tests related to the game logic, including word guessing and similar. Further unit tests could be added to test other game logic, features or connection abilities of other classes, etc.

NOTES -

- Due to the test/development environment being a single PC, all requests are made to localhost. This can be changed.
- Originally I wanted to push updates to the JSP via Comet or similar but was pressed for time. Instead the update button is used.
- Unit tests can be expanded to cover more program logic.
- Due to the potentially private nature of game logs, the logs can be accessed as a text file on the server rather than by the end user.
- **Project was developed for JDK 1.8 and Java 7 EE Web with GlassFish using Netbeans IDE 8.2**

INSTRUCTIONS FOR USE

- Run the server HangmanDealer
- Run an instance of either the basic client or the web client
- (IF ON WEB CLIENT) click “Connect” to establish a connection to the server
- Press either the 0 key (basic client) or select “New Game” and submit (web) to begin a new game
- If you wish to join an existing game, select/submit the room number (both clients)
- Type in the character you wish to guess (if you input longer strings of characters, only the first will be used)
- When the game is either failed or completed, the dealer instance will disconnect all users and terminate itself from the server, leaving a record of actions in the Log.txt file.