Unit Testing Plan

The Green Team
(Michael Philippone, Todd Wright, Matt SeGall, Erich Schudt)

SERVER-SIDE:

Server (driver class)

-> ensure that server components are instantiated
-> ensure that game play is looping until someone wins or all lose
    - make sure a move can be executed
    - make sure a guess can be executed
    - make sure an accusation can be executed
-> catch a winning / losing condition

User Creation / Management

Users / Players

-> creating 1 Player & 6 Players
-> create 7 Players and -1 Players
-> test communicating on a Player's socket
-> assign the player names
-> verify the order of Player connections
-> verify the number of Players
-> player removal

Game Engine

GameLogic

-> test guesses with all sorts of strings
    - (proper input(s), null input(s), anomalous input(s))
-> test accusations with all sorts of strings
    - (proper input(s), null input(s), anomalous input(s))
-> test movement
    - into room from hallway
    - out of a room into a hallway
    - from hallway space to hallway space
    - to room from tunnel
    - to room not using door
-> make sure that "nextPlayer" works

Pieces

-> relocating a player
    - on a "move" (user has chosen the space)

- make sure chosen space is in the “list” of possible endpoints
- on a “guess” (user has been implicated in a guess / accusation)

-> all players’ pieces end up in their appropriate start space

- print the board and verify the piece spaces

-> test if certain cell is occupied or not

-> test getting players’ positions

Cards

-> shuffle cards

-> deal the cards

-> players have cards

GameBoard

-> print out the board and verify that the image in repo (“REPO://src/clueServer/clueBoard.jpg” and “REPO://src/clueServer/clueBoard2.jpg”)

-> test if all tunnel cells are such and all non-tunnel cells are such

-> test if all doorway cells are such and all non-doorway cells are such

-> print out the serialized version of the board and verify the pieces and rooms

-> test that doorways of rooms correspond to image

Rooms

-> make sure rooms are rooms and nulls (non-spaces) are nulls and that these correspond to the “REPO://src/clueServer/clueBoard.jpg”

Dice

-> ensure that it returns random between 1 and 6 inclusive

Network

ClientSocket

-> connect to another ClientSocket instance

-> read from the socket

-> write to the socket

-> test socket closure

ServerListener

-> listen for users and receive incoming connection

-> create Players in Users class

CLIENT-SIDE

BoardModel

-> make sure that a cell can be clicked

- either makes the message for the socket or does nothing

-> can be updated with a new version

GUI

-> draw the board
-> display the cards for a user
-> get server connection info
-> guesses and accusations
-> moves
-> ignore extraneous clicks and input
-> close connections at game quit

Client

-> initiate connection with server
    - call GUI for connection info
-> start user's game
-> play through
-> handle game quit