Specifications: We are building a multiuser online whiteboard

- It can support (ignoring performance constraints) up to maxint users

- Users may edit and view one board at a time.

- The server holds a fixed number (10 for now) of boards, that are initialized at startup.

Datatype: We use two main datatypes: Brushstroke and Board. Brushtroke represents a single stroke on the canvas. It has an int start x, an int start y, an int end x, an int end y, a Color color, and an int width. They are used to draw on the canvas.

Board represents the state of the board. It has a SynchronizedList of Brushstrokes, a boardID , and a SynchronizedList of user's sockets. When something is added to the list of Brushstrokes, a message will be sent to each user.

Protocol: The server and client will communicate as follows:

-brushstroke [startx] [starty] [endx] [endy] [rgb] [width] [boardNum] – This call will be sent in both directions to represent a line being drawn. On a mouseDrag, the client will send this message to the server. The server will in turn turn the message into a Brushstroke, and add it to the appropriate board. The server will then send the same message back to each client on its user list. When the client receives this message, it will turn it into a Brushstroke, and draw it on the board via the GUI.

//-joinServer - This will be sent from client to server to join the server. The server will add the user to its user list.

//- availableBoards – This will be sent from client to server to get a list of all available boards.

//- boardList [board1] [board2] … - This will be the server's response to a availableBoards call. It will list all available boards separated by spaces.

-joinBoard [username] [boardnum] – This will be sent from client to server in order to join the board with the given boardID. The users socket will be added to the board's user list. The server will respond with a flurry of Brushstroke calls to build up the current state of the board.

-changeBoard [username] [current] [new] – This will be sent from client to server to when switching from 1 board to another.

-exitBoard [username] [boardNum] – This will be sent from client to server to have itself removed from the current Boards user list.

- getUserList [boardNumber] - This will be sent from client to server to get a list of the current users.

-userList [name1] [name2] … - This will be sent from server to client to update the clients list of users.

Concurrency Strategy: Because the whiteboard doesn't need a very high level of accuracy, we want to ensure that during concurrent editing, each stroke is recorded but we do not care about what order they are added. Since we will hold the data for the whiteboard in an array, appending to the array from

two sources should not cause issues. To ensure that our code is free of concurrency bugs, the server and client will each have thread safety arguments.

Testing Strategy: We will unit test each portion of the client and server. More specifically, on the server side, we will test the model(The list of boards, the board data type, the user list, etc.) and the controllers (The various listeners and output methods). On the client side, we will test the various listeners and the send method. Additionally, we will manually test the whiteboard to ensure that it functions as desired under actual use, with a focus on edge cases.

DESIGN NOTES:

Server:

The server will hold an arrayList of Strokes. Each Stroke represents the information needed to make one call to drawLineSegment: a start x and y, an end x and y, a color int, and a width.

The server will listen to each user on a socket, and receive a string consisting of each of these ints separated by commas, and then split it into tokens, parse those tokens into ints, and create a Stroke to add to the arrayList. The server will also have a listener attached to this arrayList, and whenever something is added to it, it will send a similar string to each client.

Client:

When the client connects, the server will send current main array line by line. Once it finishes doing so, the client will display the canvas via the GUI. The client will have a send method which will send a string consisting of the relevant integer values separated by commas to the server. The client will also be listening for similar string from the server, and when it receives one, it will call the drawLineSegment method in the GUI.

GUI:

The GUI will listen for mouseDrags, and when it hears one, it will call the send method of the client with the appropriate information. The GUI will also have a drawLineSegment method, which takes coordinates, and ints for color and thickness, and draws a line between the two points.