Design Doc

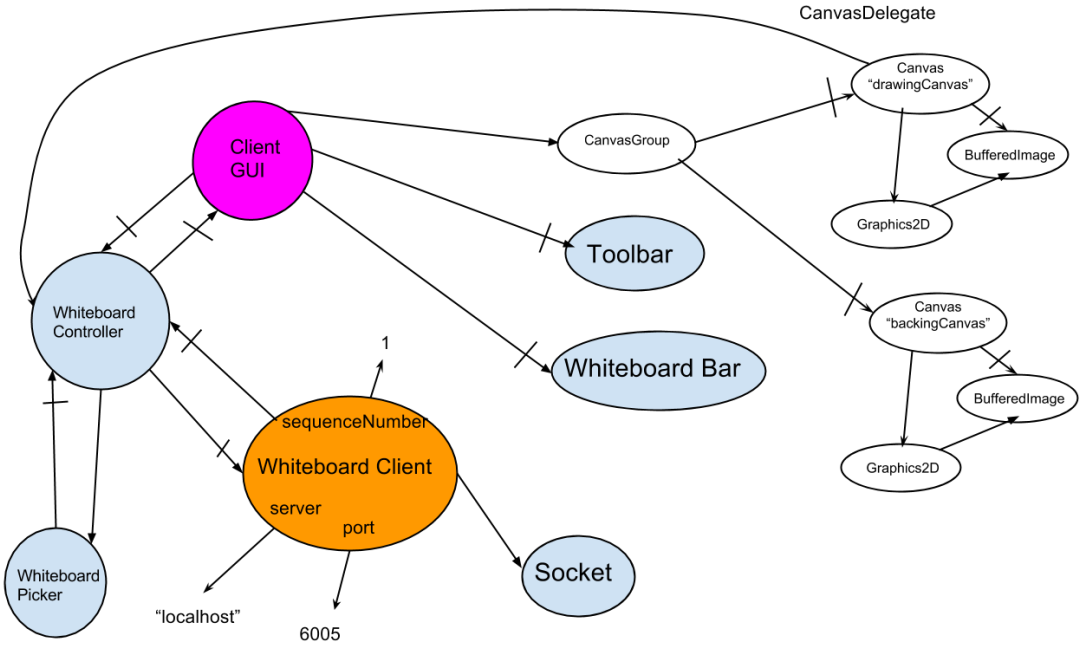
DataType Design

- Representation of a whiteboard: Colour Bitmap
- How to erase: Draw white line instead of a colour line

Classes:

- `Canvas`: Already provided, represents a whiteboard.
    - Includes a JPanel that the user can draw on
    - Includes listeners that lets the user draw
        - Listeners will also send updates to `ClientGUI`'s queue of changes
    - Includes methods to be called by `ClientGUI` to change the properties of the "brush" that draws the lines on-screen
- `CanvasGroup`:
    - A pair of Canvases layered on top of each other; one for the user to draw on and one for updates from the server.
    - The upper canvas is drawn on and used for local echo; once the server has updated the lower canvas with the user's actions (in the canonical order), the upper canvas is cleared.
- `Toolbar`:
    - This is a panel on the side with the appropriate JComponents to allow the option to:
        - change colour of the brush one of two ways
            - From a set of 8 commonly used colours
            - From a JColorChooser window
        - switch between paint and erase
        - change stroke size of the brush
        - and other yet to be decided functionality.
    - Has listeners that calls `ClientGUI's` methods in order to change the properties of the drawing tool
- `SessionBar`: This is a panel at the top which displays the current whiteboard and options to:
    - create a new whiteboard
    - pick a different whiteboard from the server
    - change the name of the whiteboard currently in use.
- `ClientGUI`: Each client runs this class as a Java application. Creates a new window that can connect to the `WhiteboardServer`.
    - One of `ClientGUI`'s purpose is to act as a wrapper for `Canvas`'s methods so that `Toolbar` can be independent of `Canvas`. `Toolbar`'s listeners call `ClientGUI`'s methods which in turn calls `Canvas`'s methods

    - A `ClientGUI` contains:
        - a `CanvasGroup` object which shows the whiteboard the client is currently working on.
        - a `Toolbar` object
        - a `SessionBar` object
    - Whenever the whiteboard is switched, the `ClientGUI` will be given a new serialized `Image` (bitmap) by the `WhiteboardServer`, representing the current state of the whiteboard
- `WhiteboardController`
    - Primary controller for the client
    - Implements all the delegate interfaces and ties together the client, picker, canvas and GUI.
- `WhiteboardClient`
    - Manages the connection to the whiteboard server.
    - Parses incoming messages for handling by appropriate handlers
    - Generates messages to send to the server
- `WhiteboardPicker`
    - User interface for selecting a whiteboard from a given list, or creating a new one
- `WhiteboardServer`:
    - Handles incoming connections and creates `Client` objects for them.
- `WhiteboardManager`:
    - Manages `Whiteboard` instances.
- `Client`
    - Represents a client, running on its own thread.
    - Receives and processes messages from the client and passes them off as appropriate to anything that is interested.
    - Equality on clients is transitive, reflexive, symmetric and consistent.
- `Whiteboard`
    - Represents a whiteboard, and contains its image data and set of active clients.
- `ClientException`
    - This is an exception. It can be thrown.
    - When it is thrown in a message handler, an error response is returned to the client.



Interfaces

- `WhiteboardClientDelegate`: defines a handler for information received from the server by the `WhiteboardClient`.
- `WhiteboardPickerDelegate`: defines a handler for actions taken by the user in the whiteboard picker.
- `CanvasDelegate`: defines a handler for actions taken by the user on the canvas (e.g. drawing lines).

All of these interfaces are implemented by `WhiteboardController`.

Protocol

- Space separated string
  - Changes to the whiteboard are represented by a colour and a series of start and end points of
- Server to client:
  - `DRAW [colour] [width] [start1] [end1] [start2] [end2] [start3] [end3]...`

        - Broadcasts updates to all clients by telling to draw the indicated lines in the indicated colour. This ensures that all changes made to the whiteboard is made known to all clients.
    - `ACK [sequence]`
        - Sent in response to a DRAW command to indicate that the command has been processed. This can be used by clients to clear local echo.
    - `JOIN [username]`
        - This message is sent to all clients on a whiteboard when a user joins it.
    - `PART [username]`
        - This message is sent to all clients on a whiteboard when a user leaves it.
    - `HELLO [whiteboards]`
        - Sent after server has established connection for a client. Sends the client a list of pre-existing whiteboards.
    - `WHITEBOARD [base64-encoded whiteboard bitmap] [names of people]`
        - Server's response client's request to join a given whiteboard. Sends the client a bitmap of what has been drawn so far on the whiteboard and a list of users currently editing this whiteboard
    - `GOODBYE`
        - Server's response to client's 'quit' request. Then disconnects the client.
- Client to server:
    - `DRAW [sequence] [colour] [width] [start1] [end1] [start2] [end2] [start3] [end3]...`
        - Lets the server know that client has drawn on the whiteboard so that the client can update all other clients connected to the same whiteboard. The drawn changes to whiteboard are in the indicated color and have the indicated start and end points. The sequence is reported back to the sending client in an ACK message.
    - `JOIN [whiteboard]`
        - This is a message requesting the server to connect this client to the selected whiteboard
    - `CREATE [name]`
        - Tells server to make a new whiteboard and switch the client to the new whiteboard. The client will also make this whiteboard available to all other clients to switch to.
    - `HELLO [username]`
        - Connects to server with given username. Username must be unique and is used by the server to keep track of the different clients
    - `QUIT`
        - Disconnects the client from the server while not changing the list of existing whiteboards

Concurrency Strategy

- Server Threading:
    - One thread per client, and one listener thread.
    - Shared resources are the WhiteboardManager and Whiteboard. All operations on these resources are performed with appropriate locking on the manager and whiteboard, ensuring that the operations are consistent and ordered.
    - *Wow. Much lock. So synchronize. Very deadlock. Too threads. Many interleaves.*
- Threading (client-side)
    - Event dispatch thread that updates the GUI. GUI window is created in invokeLater() method to make sure all GUI-related code runs on event dispatch thread and not the main.
    - Networking thread that handles messages from the server and updates the GUI in response.
    - No resources are shared between these threads, so no locking is necessary.

The client echoes the user's drawing locally on the drawing canvas. Eventual consistency is ensured by having the client track a sequence number, which is echoed back to it by the server in an ACK message. Once the client receives an ACK matching its most recent message, it clears the local echo, exposing the canonical rendition of the drawing underneath.

Testing Strategy

- Test all public methods with minimal use of client-server / threading.
- Test GUI-related behavior by inspection. Will also document testing in the appropriate classes. Test partitions:
    - Layout
        - Check to see that all the elements are where they should be, and looks as expected on different operating systems.
        - Make sure that user behavior related to the window of the application is as expected
            - Should be able to handled window resizing gracefully, the user clicking or drawing really fast
    - Responding to user's actions
        - Make sure that client draws in the same color and width as specified by click on available buttons
        - Make sure client can erase drawing (implemented as user drawing white over previous lines)
    - Concurrency

        - GUI should be responsive at all times, even when updates are being passed between server and client
        - Make sure that multiple clients can see each other's updates
-
- Server-client integration test
    - Manual tests:
        - Make sure a client can connect to server
        - Make sure server can serve multiple clients
        - Make sure that multiple clients can see each other's updates
        - Make sure that clients have the same view as each other (by some point), even if they draw over each other at the same time
        - Make sure two clients can't specify the same username
- Server-side tests
    - Manual tests:
        - Make sure client can't create new whiteboard with the same name as existing whiteboard

Features to Implement

- Show all users connected to current whiteboard
- Show list of existing whiteboards

Additional Features

- Fun Minigames
- Be able to draw different shapes (ellipses, rectangles, etc.) with different sizes
- Set mouse cursor to current tool state
- Add text