

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## DATATYPE DESIGN

- Server-side:
  - WhiteboardServer
    - \* has a list of Whiteboards
    - \* sends and receives messages to/from clients, creating and modifying Whiteboards
    - \* these messages include: selecting a whiteboard, making a new whiteboard, drawing, changing bg, users entering/exiting
  - Whiteboard (ADT)
    - \* has a unique name
    - \* holds history of all actions done to it
    - \* calculates artsy meter
- Client-side GUIs:
  - Artist (login screen):
    - \* must enter a valid IP to connect to server before anything else
    - \* then, can select whiteboard from list, or create a new one (this opens a Canvas)
      - to select an existing one, must enter username
      - to create one, must enter username, board name, bg color
  - Canvas
    - \* initially sends either a create or a select message to server, if it's a create message, we get back a history of all actions and users
    - \* sends and receives new draw actions and users connects/disconnects to/from server, displays them accordingly

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## DATATYPE DESIGN [CONTINUED]

- \* side panel provides tools that allows client-controlled adjustment
  - a pallet that holds all of Color's colors (with custom options)
  - a slider that changes the stroke size
  - a list that shows the usernames of collaborators
  - a button to clear everything from the board
- \* artsy meter at bottom of screen (whiteboard specific)
- \* rest of the window is the drawable whiteboard
- \* on close, notify server of closing

THE SECOND ROW IS VISIBLE BY DEFAULT (BECAUSE "NEW" IS DEFAULT). BECOMES INVISIBLE WHEN THE USER SELECTS AN EXISTING BOARD.

ONE OF MANY WHITEBOARDS BEING SUPPORTED AT ONCE

The illustration shows a login window titled "LOGIN" with the following fields and controls:

- ENTER USERNAME: [text input]
- BOARD: NEW (selected) [radio button]
- NAME [text input] BG WHITE: [radio button]
- ENTER IP: [text input] USE localhost [radio button]
- Go! [button]

Below the login window is a blue brushstroke representing a progress bar at 15%.

WHEN A CLIENT CONNECTS, A LOG-IN SCREEN APPEARS BEFORE THEY ARE ALLOWED TO DRAW

15% (GET ARTSIER!)

Hmm...

I wonder what this meter means!

A SLIDER TO ADJUST STROKE SIZE

DRAW!

ERASE!

CLEAR

SUPPORT FOR THE COLOR CLASS'S COLORS PLUS A MIT COLOR, AS WELL AS A COLOR PICKER. DISPLAY A PALETTE HERE.

LIST OF ARTISTS:

MR. AWESOME  
BITDIDDLES  
ANNIEJ  
MARQUEZ  
OLGASH  
6.005ROCKZ

WE'LL DISPLAY EVERYONE THAT'S CONNECTED HERE

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## METHODS

- Client (GUIs)
  - Artist
    - \* toggleNewWhiteboard(boolean visible) - controls whether new whiteboard inputs are visible. If the user does not want to make a new whiteboard, hide the enter whiteboard name and choose background color options.
    - \* addListeners() - to split the GUI into methods. This adds listeners to the buttons/text fields/combo boxes
    - \* containsSpace(String) - returns true if a String contains a space. This is to provide appropriate error popups for when usernames and whiteboardnames contain spaces.
  - Canvas
    - \* setupButtons() - separate the GUI into shorter methods; compartmentalization
    - \* addListeners() - similar to Artist;
    - \* setArtsy(int) - determines artsiness of the whiteboard using a hand-picked set of mysterious and magical criteria
    - \* addRemoveUsers(String, boolean) - adds/removes a username from the JTable of users currently working on the whiteboard
    - \* setupWhiteboard() - communications with server; Send the server the message to create/select a whiteboard, if it's a new whiteboard, nothing is returned; if it's an existing one, set the background color and users list and draw the actions.

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## METHODS [CONTINUED]

- \* `parseActions(String, boolean withArtsy)` - takes in a string representing actions and draws them. The boolean `withArtsy` lets the method know if a `Artsiness` value is expected with the string of actions
- \* `fillBackground()` - this was changed from the staff code of `fillWithWhite()`, because we would like to initialize a whiteboard with client-chosen background color
- \* `defaultSetup()` - creates a welcoming image on the default board
- \* `handleRequest(String)` - this method responds to the server's inputs by parsing the string.
- \* `paintComponents(Graphics)`, `makeDrawingBuffer()`, `drawLineSegment()`, `addDrawingController()` - these methods are provided by staff
- Server
  - Whiteboard
    - \* `makeColorList()` - a helper method for determining artsiness
    - \* `getName()` - returns a string representing whiteboard name
    - \* `addAction(8 ints)` - adds two pairs of coordinates, stroke size, and rgb values into the list of actions. Also updates the artsiness calculations
    - \* `getBackgroundColorString()` - returns the background color's RGB values in a string
    - \* `setBackgroundColor(int red, int green, int blue)` - changes the background color. This is synchronized for concurrency.
    - \* `createStringOfActions()` - returns a string representing actions performed on the whiteboard. Each action is represented by "x1 y1 x2 y2 [stroke] R G B".
    - \* `calculateArtsy()` - determines the artsiness of the board
    - \* `clear()` - clears the actions on the board

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## METHODS [CONTINUED]

- WhiteboardServer
  - \* createWhiteboard(String, int, int, int) - make a new whiteboard and add it to the list of existing whiteboards
  - \* selectWhiteboard(String, String, int) - chooses a whiteboard for according to the Client based on the board name, their username, and their client ID
  - \* createListOfActions(String) - chooses a whiteboard and calls their createListOfActions method. For descriptions, see above
  - \* listWhiteboards() - returns a string of all whiteboard names separated by spaces
  - \* listUsers() - returns all a string of all usernames separated by spaces
  - \* changeBackgroundColor(board name, RGB ints) - changes a whiteboard's background colors based on RGB values
  - \* draw(boardName, x1, y1, x2, y2, stroke, RGB) - returns a string of the draw action
  - \* clear(boardName) - clears everything from the board
  - \* putOnAllQueuesBut(int clientID, String boardName, String message) - puts the message on all the queues of clients, except the specified client
  - \* serve() - runs server, listens for and handles the connections. Has handleInput(Socket, int), handleOutput, and handleRequest as helper methods.

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## PROTOCOL

- Server - > Client
  - a list of whiteboard names (WB\_NAME WB\_NAME...)
  - lines of commands for previous whiteboard state (BGCOLOR\_R BCOLOR\_G BGCOLOR\_B ARTSY\_METER "USERS" USER\_NAME USER\_NAME... "ACTIONS" X1 Y1 X2 Y2 STROKE COLOR\_R COLOR\_G COLOR\_B X1 Y1 X2 Y2 STROKE COLOR\_R COLOR\_G COLOR\_B...)
  - new draw actions to everyone connected to a particular whiteboard ("DRAW" ARTSY\_METER X1 Y1 X2 Y2 STROKE COLOR\_R COLOR\_G COLOR\_B)
  - new client joins ("NEWUSER" USER\_NAME) to all but new client
  - change background color ("BG" COLOR\_R COLOR\_G COLOR\_B)
  - a client leaves ("BYEUSER" USER\_NAME)
- Client- > Server
  - initial connect message to request whiteboard names ("HELLO")
  - select whiteboard (add user to the whiteboard, set user name, return whiteboard state) ("SELECT" WB\_NAME USER\_NAME)
  - make new whiteboard (with color, name), like selecting, but new ("NEW" WB\_NAME COLOR\_R COLOR\_G COLOR\_B USER\_NAME)
  - new draw actions ("DRAW" WB\_NAME X1 Y1 X2 Y2 STROKE COLOR\_R COLOR\_G COLOR\_B)
  - change whiteboard bg color ("BG" WB\_NAME COLOR\_R COLOR\_G COLOR\_B)
  - disconnect message ("BYE" WB\_NAME USER\_NAME)

# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## CONCURRENCY & THREAD SAFETY

- No objects are shared between any classes, and no mutable objects shared between instances
- All messages sent are sent as strings
- No sockets shared, only IP addresses
- ID numbers for clients will use AtomicInteger as incrementing counter
- Whiteboards are treated independently and do not share information
- Draw actions and add/remove user actions are synchronized on all levels, so no information can be lost/overridden
- Drawing only happens when a message is received from the server, no local drawing, so all users on a whiteboard see the same thing
- Client and server have blocking queues that processes messages
- Only call UI repaint in SwingUtilities.invokeLater()



# DESIGN MILESTONE

ANNIEJ  
MARQUEZ  
OLGASH

## TESTING STRATEGY

- ADT tests (test public methods of Whiteboard)
- Practice drawing to make sure that the UI works as planned
- Receiving and sending commands both using local host and IP address
- Test multiple users sharing one whiteboard and multiple whiteboard support
  - make sure behavior is as expected when one user draws and another user erases common pixels
  - Multiple users drawing over common pixels
  - Strokes' real time behavior combined with erasing
  - Two strokes of different colors intersect, make sure display across clients are in sync
- Check to see that whiteboard state is saved during disconnect/reconnect

## ERROR HANDLING

- The GUI utilizes JOptionPane.showMessageDialog to provide pop-ups with specific error messages.
- Whiteboard name is taken: "That whiteboard name is taken. Please choose a different one!";
- Whiteboard name contains spaces/is empty: "Whiteboard name cannot be empty and cannot contain spaces."
- Chosen username contains spaces/is empty: "Username cannot be empty and cannot contain spaces."