Design Document Final

Project 2 - Interactive Whiteboard

Jessica Morgan Andersen

Juan Diego Castrillon

Josh Daniel Haimson

DataType Design:

Whiteboard Design:

The whiteboard consists of a Client and a ClientGUI which handle all interactions with both the user and the server. A Canvas object extends JPanel and displays a bitmap image representing the pixels of the whiteboard after a list of drawing commands have been applied to it. On the server, only the list of drawingCommands is stored so that canvases can be recreated on demand.

. Drawing commands will be sent from the client to the server and then propagated down to the canvases of all clients. The user is able to select the mode, color, and board they wish to work on. In addition, the user is able to see the other users who are also currently viewing the whiteboard.

- Mode: (Drawing, Erasing). Drawing will place a free hand draw of stroke 10 of whichever color is selected. Erasing is similar only that it will draw in white instead.
- Color: User is able to select from a color palette
- Board: User is able to select from the current open boards or select a new board.

Two users drawing simultaneously on the whiteboard will never create a problem because of the way the protocol is established between client and server. The client is only sending new drawing commands to the server while the server is pushing those changes back to each client. Thus, when two clients are modifying the whiteboard, they will each push their changes and the server will push both of those changes back to the clients.

If multiple users draw on the same pixel, the user who drew on the pixel last will show on the whiteboard. In this way, the latest user is drawing on top of the other drawings.

Protocol:

Client to Server interactions:

- New Board = "newBoard boardName"
- Switch Board = "switch username oldBoardName newBoardName"
- Exit = "exit username"
- Draw = "draw boardName command param1 param2 param3 ... "
    - Example: "draw boardName drawLineSegment x1 y1 x2 y2 color width"
- Get Users = "users boardName"
- Get boards = "boards"
- Check and add User = "checkAndAddUser username boardName"

Server to Client interactions:

- New Board = "newBoard boardName boolean"
- Switch Board = "switch username oldBoardName newBoardName command1 command2 command3..."

- Update Users = "users boardName user1 user2 user3..."
- Update Available Boards = "boards board1 board2 board3"
- Draw = "draw boardName command param1 param2 param3"
    - Example: "draw boardName drawLineSegment x1 y1 x2 y2 color width"
- Check and add User = "checkAndAddUser username boardName boolean"

Concurrency Strategy:

Client Side Concurrency:

- GUI will be made thread-safe by using Java Swing's asynchronous invokeLater method
- The local Canvas is only used as a temporary canvas to ensure that drawing is smooth and not delayed by requests from the server. Thus the master Canvas is stored on the server and any updates on the server will overwrite all updates to the local Canvas. Since all updates to the local canvas are sent to the server anyway, they will eventually propagate back down to the local Canvas. Thus, there are no concurrency issues with the Canvas or Board objects on the Client Side as they will always be overwritten by what's on the server.

Server Side Concurrency:

- The server object is made thread safe by the monitor pattern- this will ensure that no changes can be made to any of the boards while new data is being pushed out to all clients
- All methods that access the boards' rep invariants will also be protected by the monitor pattern - this ensures that two clients can't simultaneously modify a board

Deadlocks

- Since there are no methods which require nested locks, there is no chance of deadlock with this design

Race Conditions

- The precautions taken to make the server and board classes thread-safe will prevent race conditions from occurring

Testing Strategy:

View testing:

- Canvas
    - Slider sets width of stroke
    - Choosing a color from the picker sets the color of the stroke
    - Top right label
        - shows username and board name upon entering
        - updates board name upon changing boards

Integration testing:

- Client

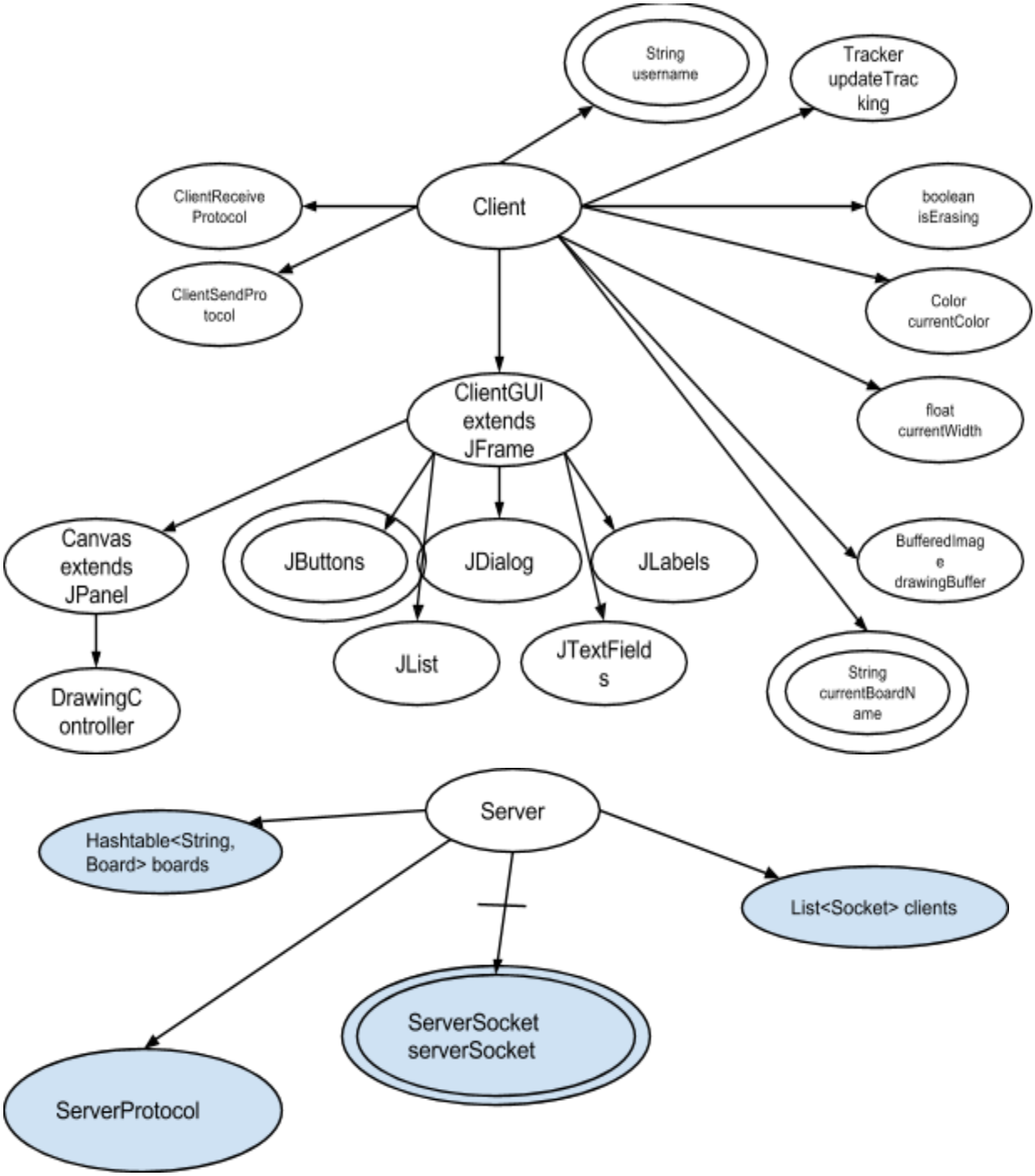
Model Testing:
- Board
- Server
- Command

Controller Testing:
- ServerProtocol (tests Server as well)
- ClientReceiveProtocol (tests Client as well)

Views will mostly be tested visually because it is hard to test the GUI using JUnit tests. However other components will be able to be tested using JUnit Tests such as the protocol. Most of the bugs will come from handling multiple users and multiple boards and concurrency; thus, extra caution must be taken into account when dealing with multiple users on a single board. Therefore we systematically tested concurrent threads.

To generate our tests, we will be partitioning our input space, and used a full Cartesian product strategy to ensure maximum coverage. We tested for bad inputs, for critical input values, and brainstormed ways that someone might try and break our code and then tested for that.

Java Classes:



- Canvas extends JPanel
    - private final Client client;
    - private EventListener currentListener;
    - public void paintComponent(Graphics g)
    - private void makeDrawingBuffer()
        - Make the drawing buffer and draw some starting content for it.
    - private void fillWithWhite()
        - Make the drawing buffer entirely white.
    - public void drawLineSegmentAndCall(int x1, int y1, int x2, int y2, int color, float width)
        - Draw a line between two points (x1, y1) and (x2, y2), specified in pixels relative to the upper-left corner of the drawing buffer.
    - public void drawLineSegment(int x1, int y1, int x2, int y2, int color, float width)
        - Draw a line between two points (x1, y1) and (x2, y2), specified in pixels relative to the upper-left corner of the drawing buffer.
    - public void updateCurrentUserBoard()
    - public JLabel getCurrentUserBoard()
    - public void addDrawingController(EventListener listener)
        - Add the mouse listener that supports the user's freehand drawing.
    - public void switchBoard(String board)
- ServerProtocol implements Runnable
    - private final Socket socket;
    - private final Server server;
    - public ServerProtocol(Socket socket, Server server)
    - public void run()
        - Waits on the client to send data then calls the appropriate request handler
    - private void handleConnection(Socket socket) throws IOException
        - Handle a single client connection. Returns when client disconnects
        - @param socket socket where the client is connected
        - @throws IOException if connection has an error or terminates unexpectedly
    - private String handleRequest(String input) throws IOException, IllegalArgumentException
    - public String boards(String[] tokens)
        - Boards response
    - public String newBoard(String[] tokens)
        - New board response
    - public String switchBoard(String[] tokens)
        - Switch board response
    - public String exit(String[] tokens)
        - Exit board response
    - public String draw(String[] tokens)
        - draw response

    - public String checkAndAddUser(String[] tokens)
        - checkAndAddUser response
    - public String users(String[] tokens)
        - Get Users response
    - public String testHandleRequest(String input) throws IOException
        - testing purposes for handleRequest()
- Server
    - private Hashtable<String, Board> boards = new Hashtable<String, Board>();
        - stores all the boards created as Board objects associated with names
    - private List<Socket> clients = new LinkedList<Socket>();
    - private final ServerSocket serverSocket;
    - public Server(int port) throws IOException
    - public void serve() throws IOException
        - Run the server, listening for client connections and handling them.
        - Never returns unless an exception is thrown.
    - public void updateBoard(String boardName, Command command)
        - Add the command on the server's queue of commands Requires valid board name
    - public synchronized boolean newBoard(String boardName)
        - Checks if the board name is unique
        - Creates a new board with the specified board name
    - public void sendCommandToClients(Command command)
        - Iterates through all the sockets and sends the command to each
    - public String getUsers(String boardName)
        - Gets the users from a board
    - public List<Command> switchBoard(String username, String oldBoardName, String newBoardName)
        - Removes the user from the old board and adds the user to the new board.
        - @param username: the username of the user making the switch
        - @param oldBoardName: name of the board the user is switching from
        - @param newBoardName: the name of the board the user is switching to
        - @return: List of Commands of the new Board the user is switching to
    - public synchronized void exit(String username)
        - Removes the user from all boards
    - public synchronized void enter(String username, String boardName)
        - Adds the user to a board for the first time
    - public synchronized String getBoards()
        - Gets a list of all the board names
    - public synchronized boolean checkUser(String username, String boardName)
        - Checks if the username is unique and if it is, return true and enter the user
    - public List<Socket> getClients()
        - Returns clients connected to server
    - public Board getCommands(String boardName)

        - Gets all commands sent to a specific board
    - public void shutDown() throws IOException
        - Shuts down all client connections and then shuts down serverSocket
    - public void addShutDownHook()
    - public static void main(String[] args)
        - Main method to launch server from command line
- Board
        - Object which represents a whiteboard stored on the server. Stores a list of all commands ever sent to whiteboard so that it can be recreated on all clients. Also stores all current users connected to this whiteboard.
    - private LinkedList<Command> commands = new LinkedList<Command>();
    - private List<String> users = new LinkedList<String>();
    - public synchronized LinkedList<Command> getCommands()
        - Returns all commands ever sent to this board
    - public synchronized void addCommand(Command command)
        - Adds a command to the board
    - public synchronized void deleteUser(String username)
        - Deletes user from board if user is in board
    - public synchronized void addUser(String username)
        - Adds user to board
    - public synchronized boolean checkUsernameAvailable(String username)
        - Checks if username is available on this board
    - public synchronized String[] getUsers()
        - Returns list of all users in board
- Command
    - private final String command;
    - private final String[] arguments;
    - private final String boardName;
    - public Command(String[] elements)
        - Creates command from token array passed that has already been determined to be a draw command
        - @param elements: Elements of command in format ["draw", "boardName", "command", "arg1", "arg2", "arg3", ...]
        - @return a Command object with the command and the arguments
    - public Command(String commandString)
        - Parses a string received from the client that has already been determined to be a draw command
        - @param commandString: the string in the format "draw boardName command arg1 arg2 arg3..."
        - @return a Command object with the command and the arguments
    - public Command(String boardName, String command, String[] arguments)
    - public void invokeCommand(Canvas canvas)
        - Finds the method with a name matching the command name, then

invokes the method with the command's arguments
    - public boolean checkBoardName(String compareBoardName)
    - public String toString()
    - public boolean equals(Object obj)
- Tracker
        - Mutable boolean object to allow tracking flags to be passed by reference
    - private boolean value;
    - public Tracker(boolean value)
        - Initializes flag to value
    - public void setValue(boolean value)
    - public boolean getValue()
- DrawingController implements MouseListener, MouseMotionListener
        - DrawingController handles the user's freehand drawing.
    - private int lastX, lastY;
        - store the coordinates of the last mouse event, so we can
        - draw a line segment from that last point to the point of the next mouse event
    - private final Client client;
    - public DrawingController(Client client)
    - public void mousePressed(MouseEvent e)
        - When mouse button is pressed down, start drawing.
    - public void mouseDragged(MouseEvent e)
        - When mouse moves while a button is pressed down, draw a line segment.
- ClientSendProtocol implements Runnable
        - Asynchronous protocol to send messages out over a PrintWriter socket
    - private final PrintWriter out;
    - private final String message;
    - public ClientSendProtocol(PrintWriter out, String message)
        - Asynchronous printwriter. Writes message to PrintWriter socket.
    - public void run()
        - Sends message to server over a PrintWriter
- ClientReceiveProtocol implements Runnable
    - private final BufferedReader in;
    - private final Client client;
    - private boolean isRunning = true;
    - public ClientReceiveProtocol(BufferedReader in, Client client)
    - public void run()
        - Waits for message from server and calls appropriate request handler
    - private void handleConnection(BufferedReader in) throws IOException
        - Handle connection to server. Returns when client disconnects.
    - private void handleRequest(String input) throws IOException, IllegalArgumentException

    - public void updateUsers(String usersMessage)
        - Checks that the board is the correct one, parses the message into an array of users and calls client.setCanvasUsers
    - public void updateBoards(String boardsMessage)
        - Checks that the board is the correct one, parses the message into an array of boards and calls client.setBoards
    - public void commandCanvas(String commandMessage)
        - Uses the Command class to create a command object and calls client.updateCanvasCommand
    - public void kill()
        - Kill thread from the outside
    - public void testHandleRequest(String input) throws IllegalArgumentException, IOException
        - testing purposes
- ClientGUI extends JFrame
    - private final Client client;
    - private final int WIDTH = 800;
    - private final int HEIGHT = 600;
    - private JDialog dialog;
    - private DefaultListModel<String> boardListModel;
    - private JLabel newBoardLabel;
    - private JTextField newBoard;
    - private JList<String> boardList;
    - private Container dialogContainer;
    - private GroupLayout layout;
    - private JTextField usernameTextField;
    - private JLabel usernameLabel;
    - private JScrollPane boardListScroller;
    - private JButton newBoardButton;
    - private JButton startButton;
    - private JFrame frame;
    - private JLabel currentUserBoard;
    - private Canvas canvas;
    - public ClientGUI(Client client)
    - public void setupCanvas()
    - private void startDialog()
        - Creates start dialog which handles username and initial board
    - public void setDialogLayout()
        - Sets layout for start dialog
    - private void setDialogActionListeners()
        - Adds action listeners to start dialog
    - public Canvas getCanvas()
    - class NewBoardWorker extends SwingWorker<Boolean, Object>

    - public void newBoardDialog()
    - private void addMenuBar()
    - private JMenu getModeMenu()
        - Add the mode menu to the menu mar
    - Add the mode menu to the menu mar
    - public void setCurrentUserBoard(JLabel newBoard)
    - private JMenu getUsersMenu()
        - Add the users menu to the menu mar
    - private JMenu getBoardsMenu()
        - Add the boards menu to the menu mar
    - private JMenu getColorsMenu()
        - Add the colors menu to the menu bar
    - private JSlider getSlider()
        - add slider to the menu bar
- Client
    - private String username;
        - the username the client will go by in this session
        - must be unique; no other clients can have this user name
    - private String currentBoardName;
        - the name of the board currently being drawn upon
    - private Color currentColor = Color.BLACK;
        - the color the user is currently drawing in
    - private float currentWidth = 10;
        - the width of the brush the user is currently drawing with
    - private BufferedImage drawingBuffer;
    - private String[] boards = {};
    - private boolean boardsUpdated;
    - private Hashtable<String, Boolean> newBoardMade = new Hashtable<String, Boolean>();
    - private Hashtable<String, Boolean> newBoardSuccessful = new Hashtable<String, Boolean>();
    - private boolean userCheckMade;
    - private boolean usersUpdated;
    - private String[] users = {};
    - private boolean exitComplete;
    - private boolean isErasing;
    - private Socket socket;
        - the socket with which the user connects to the client
    - private BufferedReader in;
    - private PrintWriter out;
    - private ClientReceiveProtocol receiveProtocol;
    - private Thread receiveThread;
    - private ClientGUI clientGUI;

- public Client(String host, int port) throws UnknownHostException, IOException
- public BufferedImage getDrawingBuffer()
- public void setDrawingBuffer(BufferedImage newImage)
- public void setIsErasing(boolean newIsErasing)
- public boolean isErasing()
- public void completeExit()
    - Confirms exit of client from server
- public void addShutdownHook()
    - Adds commands to shutdown in order to shutdown gracefully
- public boolean createUser(String username, String boardName) throws Exception
    - Checks with the server to make sure the username hasn't already been taken and if it hasn't, create the user
- public void parseNewUserFromServerResponse(String response) throws Exception
- public void switchBoard(String newBoardName)
    - Switches the current board to the board with the given name server switch command
- public void applyCommand(Command command)
- public void makeDrawRequest(String command) throws IOException
- public ClientGUI getClientGUI()
- public boolean newBoard(String newBoardName) throws Exception
    - Checks that the board name hasn't already been taken and if it hasn't, creates a new board on the server and names it with the given name
- public void parseNewBoardFromServerResponse(String response) throws Exception
- public void commandCanvas(String boardName, Command command)
    - Check that the boardName and currentBoardName are the same and then perform the command on the canvas
- public String[] getUsers() throws Exception
    - Gets the users for the current board from the server and sets them
- public String[] parseUsersFromServerResponse(String response) throws Exception
- public void setUsers(String[] newUsers)
- public Color getCurrentColor()
    - Gets the current color to use for drawing a line segment on the canvas
- public float getCurrentWidth()
    - Gets the current width to use for drawing a line segment on the canvas
- public void setCurrentWidth(float newWidth)
    - Sets the newWidth, probably based off of a slider movement on the canvas
- public void setCurrentColor(Color newColor)
    - Sets the newColor, probably based off of a color picker selection on the

canvas

- public void setUsername(String username)
- public void setCurrentBoardName(String currentBoardName)
- public String[] getBoards() throws Exception
- public String[] parseBoardsFromServerResponse(String response) throws Exception
- public void setBoards(String[] newBoards)
- public String getCurrentBoardName()
- public Thread makeRequest(String request) throws IOException
- public Canvas getCanvas()
- public String getUsername()
- public boolean checkForCorrectBoard(String boardName)
- public ClientReceiveProtocol getClientReceiveProtocol()
    - For testing purposes. Gets the ClientReceiveProtocol
- public Hashtable<String, Boolean> getBoardSuccessful()
    - Testing purposes
- public boolean getExitComplete()
    - Testing purposes\
- public static void main(String[] args)
    - Main program. Make a window containing a Canvas.