6.005 Project 2 Revised Design

by:
Grant Gunnison Nicole Brimmer Seve Esperrago

Project Overview:

This project is a server based collaborative whiteboard. Essentially a whiteboard that can logged into and drawn on from any internet connected device. The application allows for multiple users to be editing the same whiteboard from difference sources at the same time as well as multiple whiteboards to be running simultaneously. The toolset on each whitebaord allows users to specify the color they would like to draw with, the width of the tool and whether they are erasing or drawing. As long as the server is running the whiteboards are saved and can be accessed after logging in. Additionally, there is a window to view the different whiteboard that have been created as well as the users logged onto the server at any given time.

The Design below outlines what we have used to create this application. This design makes use of the MVC method and has two separate programs, the server program and the client program. The server does all of the computation, while the client program gives the user a graphical representation of the data and allows the user to create objects on their whiteboard. This design makes use of message passing between the client program and the server to update the whiteboards.

I. Datatype Design:

- **Model:** How we will represent each whiteboard in memory
    - **Class:** Point
        - **Description:** Represents a pixel in a window, each pixel in a whiteboard will be stored in memory as a Point object.
        - **Invariants:**
            - Mutable
            - Threadsafe
        - **Fields:**
            - *final int x* -- the x-coordinate of the point
            - *final int y* -- the y-coordinate of the point
            - *Color color* -- the color of the point
        - **Constructors:**
            - *public Point (final int x, final int y, final Color color)* -- creates a Point object
        - **Methods:**
            - *public int getX()*
                - returns the x-coordinate of the Point object

- *public int getY()*
  - returns the y-coordinate of the Point object
- *public synchronized Color getColor()*
  - returns the color of the Point object
- *public synchronized void setColor()*
  - changes the color of the Point object
- *public synchronized String toString()*
  - returns the String representation of the Point object

- **Class:** Whiteboard
  - **Description:** Represents a collaborative whiteboard, basically consisting of a 2D array of pixels or Point objects
  - **Invariants:**
    - Mutable
    - Threadsafe
  - **Fields:**
    - *final String name* -- the name of the whiteboard
    - *final int height* -- the height of the whiteboard in pixels
    - *final int width* -- the width of the whiteboard in pixels
    - *final Point[][] pixels* -- a width x height array of Point object, each Point object representing a pixel in the whiteboard
      - The Point object at pixels[x][y] represents the pixel at (x, y) where (0, 0) is located in the top left hand corner of the whiteboard and (width -1, height - 1) is located in the bottom right hand corner of the whiteboard.
    - *final ArrayList<String> usernames* -- the usernames of all of the client that have this whiteboard open
  - **Constructors:**
    - *public Whiteboard (final String name, final int width, final int height, final Color startingColor)*
      - creates an empty Whiteboard object.
  - **Methods:**
    - *public syncrhonized String getUsernames()*
      - returns a string containing the usernames of all of the clients that currently have this whiteboard open, separated by single spaces
    - *public synchronized String addUsername()*
      - adds a username to this whiteboard, signifying that a client has connected to this whiteboard
    - *public synchronized String removeUsername()*
      - removes a username from this whiteboard, signifying that a client has disconnected from this whiteboard
    - public synchronized boolean *hasUsername*()
      - returns boolean delineating whether or not username is in

usernames.
- *public String getName()*
  - returns the name of the whiteboard object
- *public getHeight()*
  - returns the height of the whiteboard in pixels
- *public getWidth()*
  - returns the width of the whiteboard in pixels
- *public setColor()*
  - changes the color of a pixel in the whiteboard
- public Color *getColors*()
  - returns the color of that pixel.
- *public synchronized String addLine()*
  - adds a line segment to this Whiteboard object and returns a string representation of this line
- *protected ArrayList<Point> getPointsOnLine()*
  - determines all of the points between the starting and ending points of a line segment
- *protected String colorPoints()*
  - colors all of the point specified with the specified color
- *public synchronized String toString()*
  - returns the string representation of this Whiteboard

- **Server-Client:** How we will represent clients and servers
  - **Class:** WhiteboardServer
    - **Description:** Represents the server of the Whiteboard constructors
    - **Constructors:**
      - *public WhiteboardServer(final int port)* -- creates a WhiteboardServer that listens for conenctions on port
    - **Fields:**
      - *private final ServerSocket serverSocket* -- the socket to which the server is listening for client connections
      - *private final ArrayList<Whiteboard> whiteboards* -- a threadsafe encapsulation of the whiteboards that have been created during a single session
      - *private final ArrayList<String> names* -- a threadsafe encapsulation of the names of the whiteboards thta have been created during a single session
        - The name of the ith Whiteboard object in whiteboards is the ith String in names
      - *private final HashMap<String, Socket> clients* -- the keys are the usernames of the clients that are connected to this WhiteboardServer and the values are the sockets via which each client is connected to the server

- **Methods:**
  - *public void serve()*
    - runs the server, listening for client connections and handling them.
  - *private void handleConnection()*
    - handles a single client connection. Returns when a client disconnects.
  - public synchronized boolean *checkRep*()
    - returns a boolean of whether the rep holds.
  - protected synchronized String[] *getNames*()
    - returns an array of the names of users on the server.
  - *private synchronized Whiteboard handleRequest(String name)*
    - handles a client request for a whiteboard
  - public synchronized void *sendMessageToSomeClients(final String name, final String message)*
    - send the message to all of the users that have that particular whiteboard open
  - public synchronized void *sendMessageToAllCients*(final String message)
    - sends out a message to all of the clients of the server
  - private synchronized String *createWhiteboard*(final String [] tokens)
    - returns message back to the client
  - protected synchronized String *getAllWhiteboards*()
    - returns a string with all of the names of the whiteboards
  - protected synchronized String *openWhiteboard*(final String[] tokens, final boolean notDebug)
    - returns message back to the client with the whiteboard of a specific name.
  - protected synchronized String *createUsername*(final String tokens[], final Socket socket, final boolean notDebug)
    - returns message back to the client to assign an username
  - protected synchronized String *logout* (final String[] tokens, final boolean notDebug)
    - returns message back to the client logging them out of the server
  - protected synchronized String *closeWhiteboard* (final String[] tokens)
    - returns message back to the client attempting to close the whiteboard
  - private synchronized String *getUsernamesOnWhiteboard*(final String[] tokens)
    - returns message back to the client with the usernames to a

specific whiteboard
- private synchronized String *getUsernamesOnWhiteboard*(final String[] tokens)
  - returns message back to the client
- private String *drawWhiteboard*(final String[] tokens)
  - returns message back to the server to draw a line
- private synchronized String *resetWhiteboard* (final String[] tokens)
  - returns message back to the client to clear whiteboard
- private synchronized String getAllUsernames()
  - returns a string containing all of the usernames

- **GUI:** How we will show the whiteboards
  - **Class:** Canvas
    - **Description:** This builds the mutable space for the client to write on and edit as they please. This was already supplied, but a few methods have been edited and added to.
    - Methods:
      - drawLineSegment()
        - a change was made to handle the bar that controls the width of the cursor's line.
        - a change was made to handle multiple colors.
      - setColor()
        - This was added to set the color of the line when the button on the GUI was pressed.
      - public void *addPoints*()
        - draws a series of points represented by the inputted string onto the canvas.
  - **Class:** WhiteboardGUI
    - **Description:** This builds the actual window display and the buttons that is contains. It pushes all of the computation to the canvas class to repaint the canvas.
    - Methods:
      - *public void WhiteBoardGUI()*
        - Populates the contentPane with the toolBar and canvas. It sets up the window and builds it.
      - *public void addToolBarButtons()*
        - This adds all of the buttons to the toolbar and builds the layout of toolBar.
      - public void *addActionListenerOperationOnClose*()
        - adds an actionlistener for the closing of the whiteboard window
      - public void *drawline*()
        - calls add point on the line given to add points to the canvas.
      - public void *setUsersOnlines*()

            - updates the users to show who is online.
        - public void *clearCanvas*
            - This makes the canvas completely white.
- **Class:** ClientConnectGUI
    - **Description:** This builds the window display for the client to specify what the ip address of the server.
    - **Fields:**
        - private final JLabel prompt
            - a description for the Text box.
        - private final JTextField ipAddress
            - the text box to type the ip address in
        - private final JButton connectButton
            - the button used to send the ip address.
        - private JLabel errorMessage
            - An error message will appear here if the client inputs an invalid ip address.
    - **Methods:**
        - public *ClientConnectGUI*()
            - Creates and opens a ClientConnectGUI object
        - private JPanel *createFirstRow*()
            - creates the first row of the window layout
        - private JPanel *createSecondRow*()
            - builds the second row of the window layout.
        - private void addConnectToServerActionListener()
            - adds an actionlistener to the connectButton and the text field
- **Class:** ClientService:
    - **Description:** This represents a window by which a client that has been assigned an username, can request to open and create whiteboards, and can see the usernames of the other clients that are currently connected to the server.
    - **Fields:**
        - private JTextField windowName
            - This is a text field used by the client to create the name of the whiteboard.
        - private JLabel windowLabel
            - This is the label for the text field. It tells the user what to input into the text field.
        - private JTable openWindows;
            - This is a table that shows the users online and the open whiteboards.
        - private JTextArea errorMsg;
            - This is a Text Area that allows the server to send

messages to the client to tell them about errors.
- private JButton createBoard;
  - This allows the user to create the board specified in the text field.
- private JButton openBoard;
  - This allows the user to open the board specified in the text field.
- private JScrollPane scrollPane;
  - If the table fills up with more users of whiteboards that the standard dimensions a scrollbar will appear to allow the client to see them all.
- public WhiteboardClient client;
  - This initializes the client for this particular log in.
- public UsersOnlineWindowsCreatedTable table;
  - This holds the data of which users are online and which windows have been created.

■ **Methods:**
- public *ClientService* ()
  - This sets up the all of the buttons, creates the window, creates the layout, and sets the properties of the window.
- private JPanel *createSecondRow*()
  - This creates the second row of the layout of the window.
- public void *setUsersOnLine* ()
  - This changes the table for the current users online
- public void *setWhiteboardsCreated* ()
  - Changes the table of the whiteboards that are saved on the server.
- public void *setErrorMessage*()
  - Changes the error message in the GUI window for the client to see.
- public void *addActionListenerForCreate*()
  - Adds an actionListener for the create button.
- public void *addActionListenerForOpen*()
  - Adds an actionListener for the open button.
- public void *addActionListenerOperationOnClose*()
  - Adds an actionListener for the closing of the window. This will log the client out of the server.

○ **Class:** ClientUsername:
■ **Description:** THis is a login window that allows the client to pick an username of his choice.
■ **Fields:**
- private final JLabel usernamePrompt
  - This is the label for the text box telling the client what to

enter into the text field.
        - private final JTextField username
            - This is the text field where the user inputs his username. This will be sent to the server to initialize it.
        - private final JButton createUsernameButton
            - This is a button that allows the user to send off his user name to the server and create it.
        - private JLabel errorMessage
            - This is a label that is created if there is a problem with the user name that is input into the text field.
        - private final WhiteboardClient client
            - This is what the username gets tied to. This particular client is the only one that cane hold this username.
        -
    - **Methods:**
        - public *ClientUsername*()
            - This method sets up the window, its layout, and the what the buttons read.
        - private JPanel *createFirstRow*()
            - This creates the layout of the first line of the window.
        - private JPanel *createSecondRow*()
            - This creates the layout of the second line of the window.
        - private void *addConnectToServerActionListener*()
            - This adds actionListeners to the connect buttom and the text field.
        - public void *updateErrorMessage*()
            - This updates the error message so that the client can see if there are any problems with their input.
- **Class:** UsersOnlineWindowsCreatedTable
    - **Description:**
        - This represents the table in the ClientService UI that displays the users online in the left hand column and the windows open in the right hand column
    - **Fields:**
        - private String[] *usersOnline*()
            - A list containing the usernames of all of the clients that are currently connected to the server
        - private String[] *windowsCreated*()
            - A list that contains the names of all of the whiteboards that are saved on the server.
    - **Methods:**
        - public *UsersOnlineWindowsCreatedTable*()
            - Creates an empty UsersOnlineWindowsCreatedTable

object.
- public int *getRowCount*()
  - returns the mas amount of rows in the table between the users and the windows.
- public Object *getValueAt*()
  - This returns the value at a certain row in one of the columns of the table.
- public synchronized String *getColumnName*()
  - This returns the name of the columns.
- public void *setUsersOnline*()
  - This sets the Users that are online.
- public void *setWhiteboardsCreated*()
  - Changes the contents of the “Whiteboard Created” column.

- **Class:** WhiteboardClient:
  - **Description:** The whiteboard client repersents the client. It builds the client with an username and which board is any it is connected to. This client can only have one whiteboard open at a time and the name of the white that is open in the name of WhiteBoard.
  - **Fields:**
    - private String username
      - This is the string representation of the username
    - private final ClientUsername clientUsername;
      - This is the username that is tied to the client.
    - private boolean isClientInterfaceOpen
      - boolean representing whether or not the interface is open for this client
    - private ClientService clientInterface
      - A Client Service object representing the client interface
    - private String whiteboard
      - String representation of the whiteboard.
    - private WhiteBoardGUI whiteboardGUI
      - A WhiteBoard object
    - private boolean hasWhiteboardOpen;
      - boolean representing whether or not the whiteboard is open.
    - private String[] usersOnLine;
      - list contatining all of the users that are currently logged into the server.
    - private final Socket server
      - This is the socket tied to that client.

  - **Methods:**
    - *public WhiteboardClient*()

        - This method initializes the GUI interface and starts a new thread for every connection to the GUI.
    - public String *getWhiteboardName*()
        - this returns the string whiteboard representation
    - public String *getUsername*()
        - returns the string representation of the username
    - protected void setUsername()
        - assigns the username to this specific instance of this whiteboard client.
    - protected synchronized void openClientInterface()
        - This opens the client interface which allows the client to request whiteboards to be opened and created.
    - public void *sendMessage*()
        - communicates with the server by sending specific messages to the server

II. Message Passing Protocol:

create [name] -

If [name] contains any spaces, this method returns an output message of the form "Whiteboard names cannot contain any spaces."
If a whiteboard whose name is [name] is saved on the server (i.e. has already been created), this method returns an output message of the form "A whiteboard with that name has already created. Please choose another name."
Otherwise, if a whiteboard whose name is [name] is not saved on the server (i.e. has either never been created or was deleted) and [name] does not contain any spaces, this method creates a whiteboard whose name is [name] and returns the name of the whiteboard this method also notifies all other clients of this change with a message of the form all usernames [whiteboard 1] [whiteboard 2] [whiteboard 3] ... [whiteboard n] where [whiteboard 1] [whiteboard 2] [whiteboard 3] ... [whiteboard n] are the names of all of the whiteboards on the server

open [username] [name] -

[username] must represent the client that is connecting through the currently handled port.
If a whiteboard whose name is [name] is saved on the server, this method returns a string representation of the whiteboard to the client and adds the client to the specified whiteboard this method also notifies all clients currently connected to the specified whiteboard of the new client
Otherwise, then returns an output message of the form "A whiteboard with that name does not exist."

logout [username] -

[username] must represent the client that is connecting through the currently handled port.
This method returns an empty output message and removes the client from all whiteboards that it is connected to and notifies all other clients both of the fact that they have disconnected from the whole server and that they have disconnected from all of the whiteboards

username [username] -
If a client whose username is [username] is connected to the server, this method returns "That username is already being used."
If [username] contains a space, this method returns "There should be no spaces in the username."
Otherwise, this method creates an entry in the HashMap corresponding to this username and client and returns the requested username

getUsersOnWhiteboard [whiteboard]
This method returns "alsoediting [usernames]" where [usernames] is a String containing the usernames of all clients currently editing the whiteboard whose name is [whiteboard].
[whiteboard] must be the name of a whiteboard that is saved on the server

list -
This method returns "allwhiteboards [whiteboards]" where [whiteboards] is a String containing the names of all the whiteboards currently saved on the server, in the order that they were created, each separated by a space

reset [whiteboard]
[whiteboard] must be the name of a whiteboard that is saved on the server and sends a message of the form "reset" to all of the clients connected to the whiteboard [whiteboard]
This method clears the whiteboard whose name is [whiteboard].

close [username] [name] -
[username] must a client on the server that is connected to whiteboard
[name] must be a whiteboard on the server
This method removes the client whose [username] from whiteboard whose name is [name]

draw [name] [x1] [x2] [y1] [y2] [red] [green] [blue] [thickness] -
[name] must be a whiteboard on the server
This method requests a line to be draw on the whiteboard named [name] from ([x1], [y1]) to ([x2], [y2]) with color [red] [green] [blue] and thickness [thickness]
Note that ([x1], [y1]) and ([x2], [y2]) must be points on the whiteboard named [name]
This method returns an empty string.
This method also sends out a message to all of the clients connected to the whiteboard [name] that this line has been added to the whiteboard

III. Concurrency Strategy:
The classes Point and Whiteboard are both threadsafe (as indicated in the section Datatype Design). The Whiteboard object is synchronized on Point objects so that each multiple servers can change different parts of the same Whiteboard object at the same time, but they cannot

change the same part of the same Whiteboard object at the same time.

There is main thread for the server and a different thread handling each of the clients. Each client will also has a GUI thread handling the whiteboard display window.

IV. Testing Strategy:

- 3 Levels of Testing
    - Unit Testing
    - Testing via Isolation/Hardcoding
    - Integrated Process
- Unit Testing
    - Point
        - JUnit testing of datatype methods
    - Whiteboard
        - JUnit testing of datatype methods
    - ClientService(GUI)
        - manual testing
            - Create a valid user:
                - Addition of username into table
            - Request to create an invalid board:
                - The board name has a space.
                - The board name has already been added.
                - The board name uses characters other than ascii characters.
                - The length of the board name is greater than Integer.MAX_INT
            - Request to create a board by:
                - Entering board name into textfield,
                - Clicking on 'Create Board' button.
            - Request to open an invalid board:
                - The board name has not already been added.
            - Request to open a board by:
                - Entering board name into textfield,
                - Clicking on 'Open Board' button.
            - Creation and opening of boards is case-sensitive.
            - Client is only allowed to open one whiteboard at a time.
    - ClientUsername (GUI)
        - manual testing
            - Request a username that is not valid:
                - The username has a space.
                - The username has already been added.
            - Request a username by
                - clicking on the button.

            - pressing enter in the text box.
    - ClientConnectGUI
        - manual testing
            - Enter an IP address that is valid.
                - It should connect and close itself.
            - Enter an IP address that is not valid.
                - It should not connect and instead an error message should pop up on the GUI.
            - Enter an IP address by
                - clicking on the button "Connect!"
                - pressing enter in the text box.
            - Enter same IP address concurrently as other user
                - should be able to connect to the same server
    - WhiteBoardGUI
        - manual testing
            - Click each button draw on canvas with each respective setting.
                - Draw
                - Erase
                - Erase All
                - Red
                - Orange
                - Yellow
                - Green
                - Blue
                - Pink
            - Move around slider bar to change thickness.
            - Concurrent Users:
                - Drawing at same time in different places
                - Drawing at same time in the same place with different colors
                - Drawing and erasing at the same time
                - Drawing with different thickness in different place
                - Drawing with different thickness in the same place
                - Drawing and erasing all at the same time
                - Showing the usernames of the users that are currently editing the board
- Testing via Isolation/Hardcoding
    - WhiteboardServer
        - Hardcoded messages being passed in and making sure the server reacts in the correct manner in each instance
            - (1) create [name]
                - JUnit and manual testing
            - (2) open [username] [name]

            - JUnit and manual testing
        - (3) logout [username]
            - JUnit and manual testing
        - (4) username [username]
            - Manual testing
        - (5) close [username] [name]
            - Manual testing
        - (6) draw [name] [x1] [y1] [x2] [y2] [red] [green] [blue] [thickness]
            - Manual testing
        - (7) list
            - Manual testing
        - (8) getUsersOnWhiteboard [whiteboard]
            - Manual testing
        - (9) reset [whiteboard]
            - Manual testing
        - Invalid command
    - WhiteboardClient
        - Hardcoded messages being passed in and making sure the clientside is reacting in the correct manner - done through manual programming
            - (1) "usernameerror That username is already being used."
            - (2) "usernameerror There should be no spaces in the username."
            - (3) "usernameCreated [username]"
            - (4) "allUsersOnline [usernames]"
            - (5) "whiteboardnameerror Whiteboard names cannot contain any spaces."
            - (6) "whiteboardnameerror A whiteboard with that name has already created. Please choose another name."
            - (7) "whiteboardcreated [name]"
            - (8) "allwhiteboards [whiteboards]"
            - (9) "open [whiteboardname] [whiteboard]"
            - (10)"alsoediting [usernames]"
            - (11)"whiteboardopenerror A whiteboard with that name does not exist."
            - (12)"drawLine [line]"
            - (13)"reset"
            - Invalid command
- One Test To Rule Them All (IntegratedTest)
    - Initialize server and run the server.
    - Initialize 2 clients
    - Create 1 whiteboard to be shared amongst each other
    - Each client edits the whiteboard
        - Each client drawing a line
    - Client 1 resets the board