Damon Doucet, Harini Kannan and Kulpreet Chilana
6.005 Software Construction
December 2, 2013

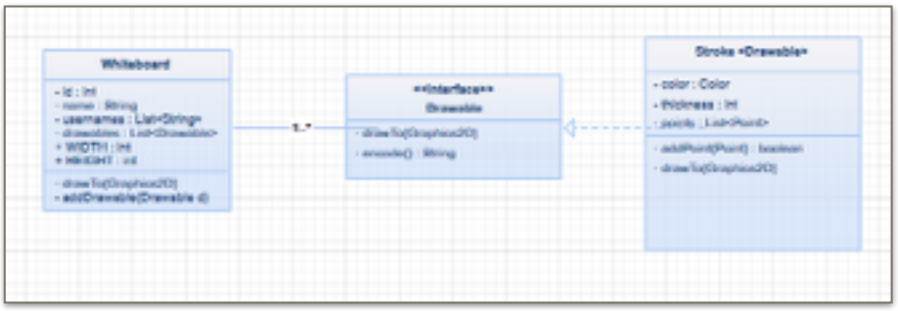
Preliminary Design
Whiteboard Project

Datatype Design

Domain

- **Drawable Interface**
  - drawTo(Graphics2D)
- **Whiteboard**
  - Properties
    - ID
    - Name
    - List<Usernames>
    - List<Drawable>
  - Constants
    - Width
    - Height
  - Methods
    - drawTo(Graphics2D)
      - calls drawTo on the list of Drawables with the same Graphics2D object
    - addDrawable(Drawable)

- **Stroke** IMPLEMENTS **Drawable**
  - Properties
    - Color
    - Width
    - List<Point>
  - Methods
    - addPoint(Point)
    - drawTo(Graphics2D)



- Future Ideas: Rectangle, Ellipse, etc (IMPLEMENTS Drawable)

Client

- Networking
  - Grammar
  - Sockets
- Login Window
- Whiteboard Menu Window
- Drawing Window
  - Tool Panel
  - Actual Drawing Board

Server

- Whiteboard List
- Individual Whiteboard Updates
- Networking
  - Grammar
  - Sockets
- Basics
  - Threading
  - Application

Networking Protocol

Actions

- **Login**
  - Client must input username
- **Menu**
  - Client should know when a new whiteboard is created
    - Should it know the creator's username?
    - Should it know the names of all who are connected?
    - Whiteboards are never deleted
  - Client can join a whiteboard
  - Client can create a whiteboard with a given name
    - Names are not necessarily unique
- **Drawing**
  - Client can draw
  - Client can return to menu
  - Client should know when people join/leave

Grammar

```
General
      NICKNAME := [a-zA-Z0-9]{2,5}
      INT := [1-9][0-9]*
      NEWLINE := "\r"?   "\n"
      BOARD := BOARD_ID "-" BOARD_NAME
            BOARD_ID := INT
            BOARD_NAME := [a-zA-Z0-9][a-zA-Z0-9 ]{2,10}
      drawing_action := "DRAW" stroke NEWLINE
            stroke := "STROKE" COLOR THICKNESS POINT{2,}
            THICKNESS := INT
            COLOR := "WHITE" | "BLACK"
            POINT := INT "," INT

Client to Server
      Login
            nickname := "NICK" NICKNAME NEWLINE
      Menu
            menu_action := [ make | join ]
            make := "MAKE" BOARD_NAME NEWLINE
```

```
            join := “JOIN” BOARD_ID NEWLINE


      Drawing
            on_draw := drawing_action
            leave := “LEAVE” NEWLINE
      Exit
            bye := “BYE” NEWLINE

Server to Client
      Login
            specify_nick := “SPECIFYNICK” NEWLINE
            nick_in_use := “NICKINUSE” NEWLINE
            nick_ok := menu_list NEWLINE
      Menu
            menu_list := “MENU” BOARD* NEWLINE
            board_created := “NEW” BOARD NEWLINE
            id_not_found := “BADID” NEWLINE
      Drawing
            on_join
                  sends  a list of on_friend_draw and on_friend_join messages
            on_leave := menu_list
            on_friend_join := “LEAVE” NICKNAME NEWLINE
            on_friend_leave := “JOIN” NICKNAME NEWLINE
            on_friend_draw := drawing_action
```

Concurrency Strategy

Client

- GUI and Socket thread
- They won't interact. The socket thread handles reading and writing to the socket and has a callback to the GUI thread for drawing
    - The socket has a threadsafe "send message" function for the client to call
    - The callback on the GUI will use invokeLater to safely draw parsed messages

Server

- Separate thread for each client, and a single thread for the server.
- The MessageBus and queueing mechanism allow all of these to communicate thread-safely
    - The MessageBus pushes messages onto necessary thread-safe queues which the client/server thread will read on their own time

Testing Strategy

Domain

- Models
    - We will use unit tests to test every single method of our datatypes.
    - In the Stroke class, we will test drawTo, addPoint, encode, encodeColor, and encodePoint.
    - In the Whiteboard class, we will test getId, getName, getDrawables, drawTo, addDrawable, signInUser, and signOffUser.
        - Note that drawTo will be tested manually.
- More details about Encoding/Decoding test case partitions
    - Decoder
        - Valid inputs
            - Two points
            - Lots of points
            - Black and white
            - Varying thicknesses
        - Invalid inputs
            - Missing parameters
            - Not enough points in stroke
            - Bad colors
            - Floating point thickness
            - Point missing a parameter
    - Encoder
        - Decode(Encode(x)) == x
        - Several general cases

Client

- The Client code will be decoupled into separate GUI objects for each type of window and the networking client, allowing us to test each component separately.
- GUI
  - The GUI will be tested manually for the following test case partitions:
    - A single user being able to draw lines on the whiteboard with various colors
    - A single user being able to erase previous strokes on the whiteboard
    - Multiple users trying to draw on the same spot of the whiteboard at the same time
    - Two separate users trying to draw and erase at the same spot of the whiteboard at the same time
- Networking
  - The main functionality to ensure is that the client can correctly parse the incoming messages from the server using the grammar.
  - We will be testing for the following test case partitions in the grammar: login, menu, drawing, and exit (see Grammar section for more details)
    - We will ensure the messages are parsed as they should be (according to the grammar)

Server

- MessageBus
  - Test publish/subscribe
  - Test hasKey for whiteboards + clients
- ClientHandler State Machine
  - Test state transitions by feeding input and watching state of the given ClientHandler
    - We can pass a null Socket to the ClientHandler since the socket is only referenced in run() (when the thread is started)
  - Test invalid input
- Server State

End-To-End Tests

- Run the server
- Ensure multiple clients can connect
- Ensure the same nickname is not supported
- Ensure that creating a board on one shows for the others
- Ensure that joining/leaving a board updates connected users for both
- Ensure that drawing on the board updates for the other
- Ensure that joining the board after drawing has happened will show correct picture
- Ensure that creating another board is independent in both connected users and drawing of the other
- Ensure that everyone disconnecting from the server and then rejoining still has the same whiteboard

Incidentally, several of these end-to-end tests can (and should) also be written as unit tests.

Additional Considerations

- What set of editing actions are provided
  - Eraser - white strokes
  - Strokes
    - Must support multiple colors
  - Eventually other shapes like rectangles and circles?
- How the whiteboard is structured
  - List of Strokes
    - index of a stroke is its z-index (ordered by time)
- How whiteboards are named and accessed by users
  - Server maintains a list of Whiteboards
    - has connected users
    - has a title
  - When a Client initially connects, they see a menu of whiteboards
    - Whiteboards cannot be deleted
    - Can create a new whiteboard
      - Prompted for a name
      - Names do not have to be unique
    - Can join an open whiteboard
      - Name is shown
      - When a new whiteboard is created, all users should be notified
    - Can return to the menu of open whiteboards from the drawing part
- How whiteboards are stored or cached (e.g. at a central server or on clients)
  - Central server
- What guarantees are made about the effects of concurrent edits
  - Server will process them in the order it gets them
  - Since the whiteboard is a list of strokes, the only issue with concurrency might be the z-indices
    - At first, we will accept this as a reasonable issue
    - If we have time later, we'll fix it

Designing architecture and protocol.

1. You must also devise a network architecture and a protocol for this project.
2. Architecture - client/server architecture
3. Grammar - a text-based protocol
4. Using sockets just like in the Minesweeper problem set

Handling multiple users.

- Server
  - Server Queue
  - Threadsafe Queue of messages to be sent to each client
  - Server Thread
    - Polls Server Queue of messages to add to shared state
    - Pushes messages onto client-specific queues when messages need to be sent
  - Thread for each client
    - Polls its specific queue for messages to be sent
    - Adds messages to server queue
  - One final thread that handles incoming
- Client
  - GUI thread
  - Server connection thread

Additional Features

Things we might consider doing, time-permitting:

- Menu
  - Shows all people connected to whiteboards
  - Shows thumbnail of whiteboards
- Drawing
  - Other shapes (rectangles, ellipses, etc)
  - Additional colors
  - Stroke widths
  - Possibly more realtime