Design Document
bschang, kellyliu, krwang

Datatype Design:designing how whiteboard functions for client

WhiteboardGUI:

- Purpose: this class combines the canvas and tools panel in one jFrame to present a unified appearance for the GUI
- it will also keep track of the current users and display their usernames. the users will be stored in a Set, so there can't be repeat usernames
- public WhiteboardGUI(): constructs a WhiteboardGUI
- Note that User methods and instance variables will be altered later on to accommodate for multiplle users on same whiteboard--should also have a user instances

Canvas:

- Purpose: this class represents the actual surface upon which the user can draw with their mouse. It also stores the image on the screen, which will be saved so that when the user reaccesses that specific instance of the canvas, the image will be saved with it
    - this image is a 2d array of pixels
- public Canvas(int width, int height): creates a canvas of a designated width and height
- public void paintComponent(Graphics g): the method in which the drawing buffer is actually drawn on the screen

ToolsPanel:

- Purpose: Allows users to customize their current drawing settings (brush type, brush size, brush color)
- Part of the overall GUI
- public ToolsPanel(): constructs a ToolsPanel
- All functions contained in action listeners, as the panel responds to user actions to customize their brush settings

User:

- Purpose: this separates the user's data from the toolspanel so that the toolspanel only keeps track of the GUI and the user's data is isolated to this class
- public User(ToolsPanel toolsPanel)
- public static void setCurrentBrush(int newBrush)
- public static void setCurrentSize(int newSize)
- public static void setCurrentColor(int newColor)
- public static int getCurrentBrush()
- public static int getCurrentSize()
- public static int getCurrentColor()
- protected????

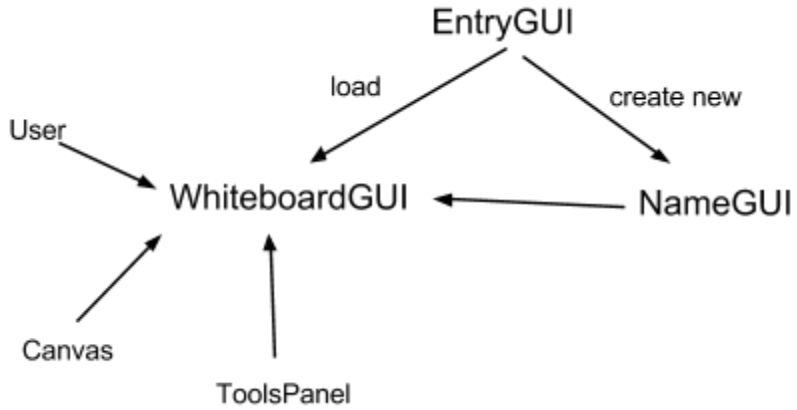
EntryGUI:

- Purpose: GUI for users to specify whether they want to create a new Whiteboard on the server or load a previously created Whiteboard from the server
- When creating a new Whiteboard, the GUI prompts the user to enter a name for the Whiteboard that the server will store the Whiteboard under.
- Contains a definition and constructor for a NameGUI instance, which is the prompt for the user to input a name
- public EntryGUI(): constructs a EntryGUI

Server:

- We will have the server contain a hash map that will contain all the created canvases. The hash map will map from the Canvas name to its contained Image. Users will be able to load Canvases from the hash map by inputting the name of the Canvas into the load text field in the EntryGUI
- Canvases must have unique names, so as to prevent any potential overwriting or duplicate entries in the hash map
- public Server(): creates a new Server instance
- public Canvas load(String): Pulls a preexisting Canvas from the hash table using the passed in String
- public boolean save(Canvas): Updates the current entry in the hash table with the updated Canvas
- public Canvas makeNew(String): Creates a new Canvas instance and stores it in the hash table
- public String getUniqueName(): Returns a unique name identifier for a Canvas

Snapshot Diagram:



Protocol: designing how client and server interact

- We will not be using telnet because we see no added benefit to debugging in this fashion. Because we will not be using telnet, we do not need to create a grammar for this client/server protocol
- The client requests a pre-existing or new canvas instance from the server, and the WhiteboardGUI automatically sends updated canvas instances to the server after specific

time intervals (autosave) and after all users exit a whiteboard "room"

Concurrency Strategy:

- We would like to have all users be able to draw on whatever whiteboard they choose concurrently
- There cannot be any deadlocks because users are not dependent on other users, there is no dependency between classes
- Another concurrency problem would occur if two users are trying to draw on the same whiteboard at the same time because the users could have different settings with which they're drawing
- If two users are trying to draw on the same whiteboard at the same time at the same location, we will allow the race condition to perform as is because it doesn't really matter who would draw first (??????????????)
- We will employ a client-server design implementation along with a confinement strategy to prevent multi-user access to anything but the shared Canvas object, which will in turn ensure thread-safe behavior for all datatypes aside from the Canvas class (the thread safety argument for the Canvas class is described above)

Testing Strategy

- **Helper Classes:** will test each method with JUnit tests
- **GUI:** document the possible interactions with the GUI and use println to display what the GUI reads as the action and the response. We will run the GUI and tests each possibility.
    - test one user one whiteboard--test every type of brush, every color, every size brush
    - test one user multiple whiteboards: drawing on multiple whiteboards to make sure drawing are accurate
    - test multiple user same board: drawing at the same time, drawing at different times, drawing at the same location at the same time, and using different brushes/colors/sizes all of these times
    - test multiple users multiple whiteboards
    - test someone drawing on a Whiteboard and someone erasing from the same spot on the same Whiteboard at the same time (indeterminate result???)
- **Server:** Using JUnit tests we will test all messages that can be passed to the server and check that the server returns an appropriate response.
    - test opening/closing/loading a Whiteboard
    - test opening/closing/loading multiple Whiteboards (make sure the server keeps track of which edits are performed on which Whiteboards)
    - test one user editing a Whiteboard
    - test multiple users editing one Whiteboard (checking for race conditions?)
    - test multiple users editing multiple Whiteboards (make sure the server keeps track of which edits are performed on which Whiteboards)
    - testing having up to 3 users editing one Whiteboard
    - test accessing a whiteboard that has already been created in the past

We are essentially trying to test every single aspect of our project, so this amount of effort should

be enough. We are also testing all potential problems with each aspect of our project.

Handling multiple users. Since realtime collaborative whiteboard is not realtime without at least two people, your system must be able to handle multiple users connected at the same time. One reasonable design approach could be using one thread for reading input from each client and having a central state machine representing the state of the server (using one more thread, to which each of the client threads pass messages through a shared queue).

clients send data to their thread, which sends the output to a central thread

concurrent reading in input from all clients--bc each client is on diff thread, so you can do it at the same time

each whiteboard will be concurrent

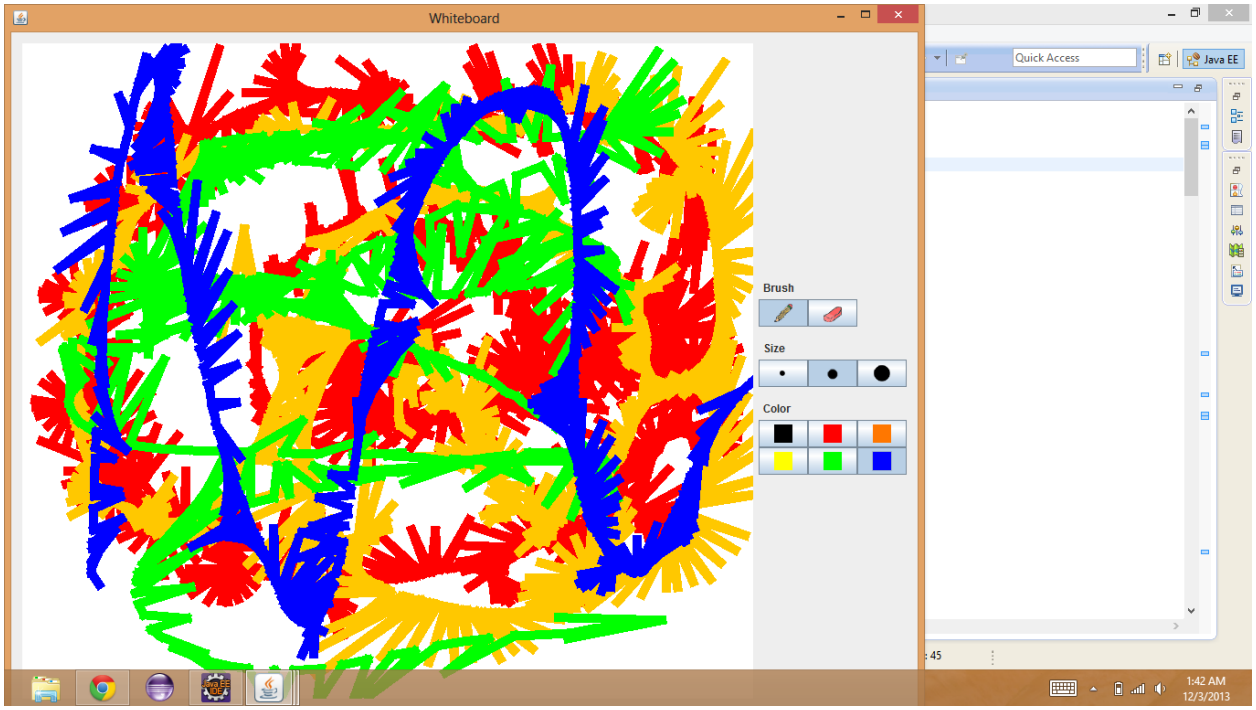
for each whiteboard, there will be one thread that all inputs get put into so single file through events/actions

testing canvas:
create byte by byte representation of each pixel for the console for testing
test location of drawing
test length of drawing
test erase function
test color change
test brush stroke size change
test drawing a line segment
test opening/closing canvas
test loading a canvas

test drawing in same location as another user
drawing on different whiteboards at the same time
open and close board and data still stays
create new board
open past board and same results on it as before
different users edit same board--all changes saved

--open (new one), --close, --load (existing one), --brush -size -color, --erase, --draw

- What set of editing actions are provided
- How the whiteboard is structured
- How whiteboards are named and accessed by users
- How whiteboards are stored or cached (e.g. at a central server or on clients)
- What guarantees are made about the effects of concurrent edits


Whiteboard
Brush
Size
Color
Quick Access
Java EE
1:42 AM
12/3/2013