Pass The Drawing!

A turn-based multiplayer drawing game written in Java

Martin Lonnemann

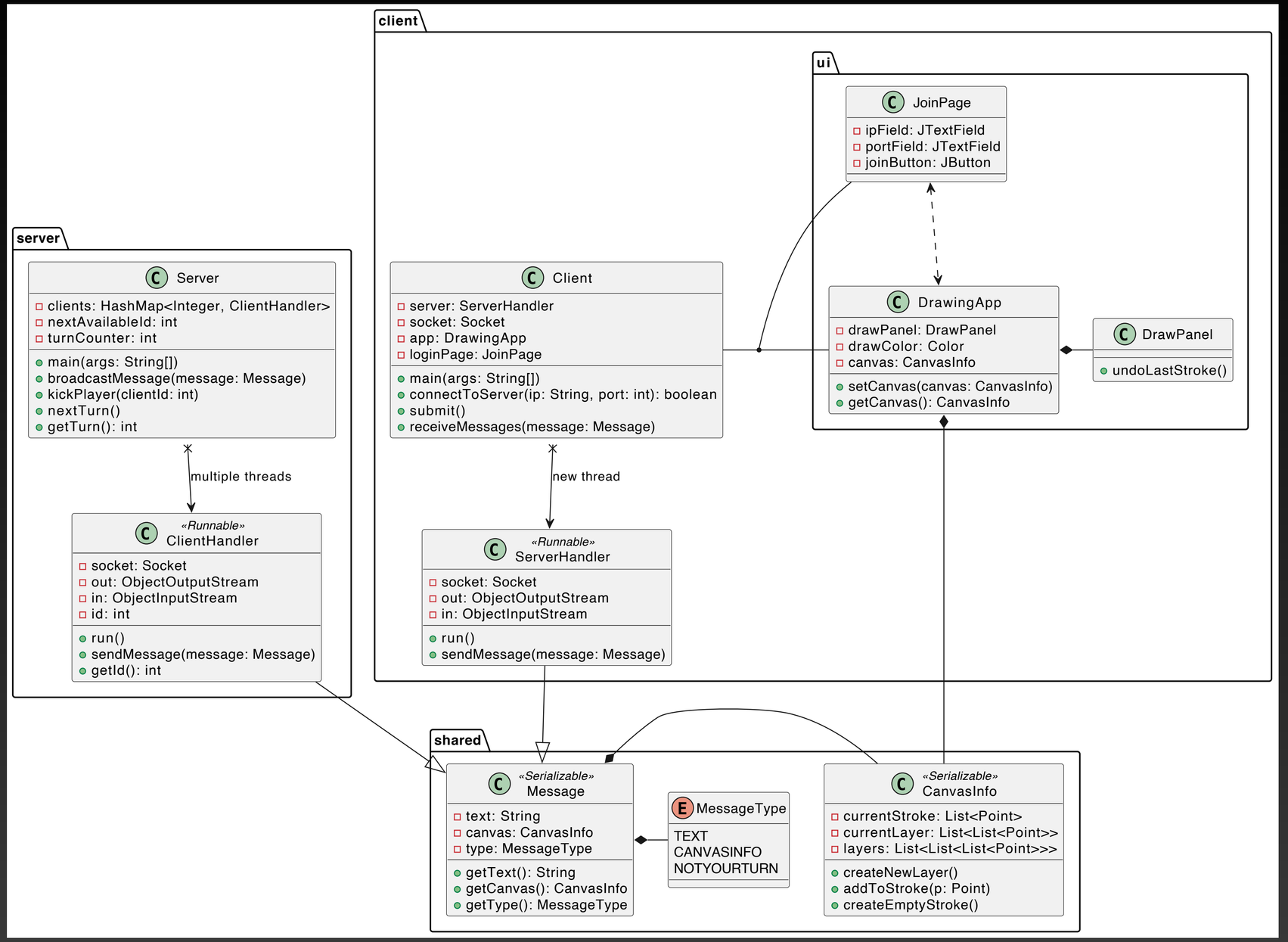
The Game

- Multiple users connect to a server
- If it is your turn, you may draw on a canvas.
- After you submit your part of the drawing, the drawing is then passed on to the next person.
- This rotation continues until you have a beautiful masterpiece

Technical Concept

- Server
  - Waits for incoming clients and spawns a new thread for each connection
  - Holds game information, like who’s turn it is
- Client
  - Spawns a new thread to handle the server connection
  - A GUI is built using the Swing library
  - The GUI has a drawing canvas, allowing the player to draw

UML Diagram


client
ui
JoinPage
ipField: JTextField
portField: JTextField
joinButton: JButton
server
Server
clients: HashMap<Integer, ClientHandler>
nextAvailableId: int
turnCounter: int
main(args: String[])
broadcastMessage(message: Message)
kickPlayer(clientId: int)
nextTurn()
getTurn(): int
multiple threads
«Runnable»
ClientHandler
socket: Socket
out: ObjectOutputStream
in: ObjectInputStream
id: int
run()
sendMessage(message: Message)
getId(): int
Client
server: ServerHandler
socket: Socket
app: DrawingApp
loginPage: JoinPage
main(args: String[])
connectToServer(ip: String, port: int): boolean
submit()
receiveMessages(message: Message)
new thread
«Runnable»
ServerHandler
socket: Socket
out: ObjectOutputStream
in: ObjectInputStream
run()
sendMessage(message: Message)
DrawingApp
drawPanel: DrawPanel
drawColor: Color
canvas: CanvasInfo
setCanvas(canvas: CanvasInfo)
getCanvas(): CanvasInfo
DrawPanel
undoLastStroke()
shared
«Serializable»
Message
text: String
canvas: CanvasInfo
type: MessageType
getText(): String
getCanvas(): CanvasInfo
getType(): MessageType
MessageType
TEXT
CANVASINFO
NOTYOURTURN
«Serializable»
CanvasInfo
currentStroke: List<Point>
currentLayer: List<List<Point>>
layers: List<List<List<Point>>>
createNewLayer()
addToStroke(p: Point)
createEmptyStroke()

Threading

- On the server, when a client connects, we create a new socket and spawn a handler thread, that uses that socket to communicate with each individual client.
  - This ensures we can have multiple clients interacting with the server.
  - We add a reference to each ClientHandler object into an array, so we can access its methods from a global context.

Synchronization

- The `synchronized` block accesses an object's intrinsic lock or **mutex**, ensuring mutual exclusion
- All methods that interact with game data are labeled as `synchronized`, only allowing one thread to interact with the game data at a time.
- During a synchronized method execution, a lock is acquired on the object and released when its done executing OR when an exception is thrown.

Shared Data Structures

- The Message class is an object that holds various data to pass between the server and the client.
- On creation of a new Message, a MessageType (enum) is declared for what message it is.
  - TEXT, CANVASINFO, and NOTYOURTURN.
  - This is useful for parsing the message on both the Client and the Server Side
    - If the message is of CANVASINFO type, we know that the message object holds the drawing information that is sent from a client. (CanvasInfo object)
- Both the Message class and CanvasInfo objects extend the **Serializable** class, allowing for the object to be **converted into bytes to be sent over the socket.**

Sources

- **Critical Sections (Synchronization):**
  - https://www.baeldung.com/java-synchronized
  - https://docs.oracle.com/javase/specs/jls/se21/html/jls-17.html
  - https://docs.oracle.com/javase/specs/jls/se21/html/jls-14.html#jls-14.19
  - https://docs.oracle.com/javase/specs/jls/se21/html/jls-8.html#jls-8.4.3.6
- **Sockets**
  - https://www.geeksforgeeks.org/socket-programming-in-java/
  - https://docs.oracle.com/javase/8/docs/api/java/net/ServerSocket.html
  - https://docs.oracle.com/javase/8/docs/api/?java/net/Socket.html
  - https://docs.oracle.com/javase/8/docs/api/java/io/ObjectInputStream.html
  - https://stackoverflow.com/questions/26320156/using-buffered-streams-for-sending-objects
  - **Server with Multi-Client Interaction**
    - https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/
    - https://github.com/oreillymedia/java_cookbook_3e/blob/master/javacooksrc/javacooksrc/main/java/network/EchoServerThreaded2.java
- **Multiprogramming**
  - https://www.w3schools.com/java/java_threads.asp
  - https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html
  - https://download.java.net/java/early_access/valhalla/docs/api/java.base/java/lang/Thread.html
  - https://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/ThreadPoolExecutor.html
- **UI**
  - https://docs.oracle.com/javase/tutorial/uiswing/index.html
  - https://www.geeksforgeeks.org/java-jframe/
  - https://docs.oracle.com/javase/8/docs/api/index.html?javax/swing/JFrame.html
  - https://docs.oracle.com/javase/8/docs/api/javax/swing/JPanel.html
  - https://docs.oracle.com/javase/8/docs/api/javax/swing/JComponent.html#paintComponent-java.awt.Graphics-
  - https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/java/awt/event/MouseMotionListener.html
  - https://docs.oracle.com/javase/8/docs/api/index.html?java/awt/event/MouseAdapter.html