Project 6
Peer to Peer Two-Player Gomoku (Five in a row)

Aishwarya Paithankar
Varad Raut

November 16, 2022

Status Summary

Title: Peer to Peer Two-Player Gomoku (Five in a row)
Team Members: Aishwarya Paithankar, Varad Raut

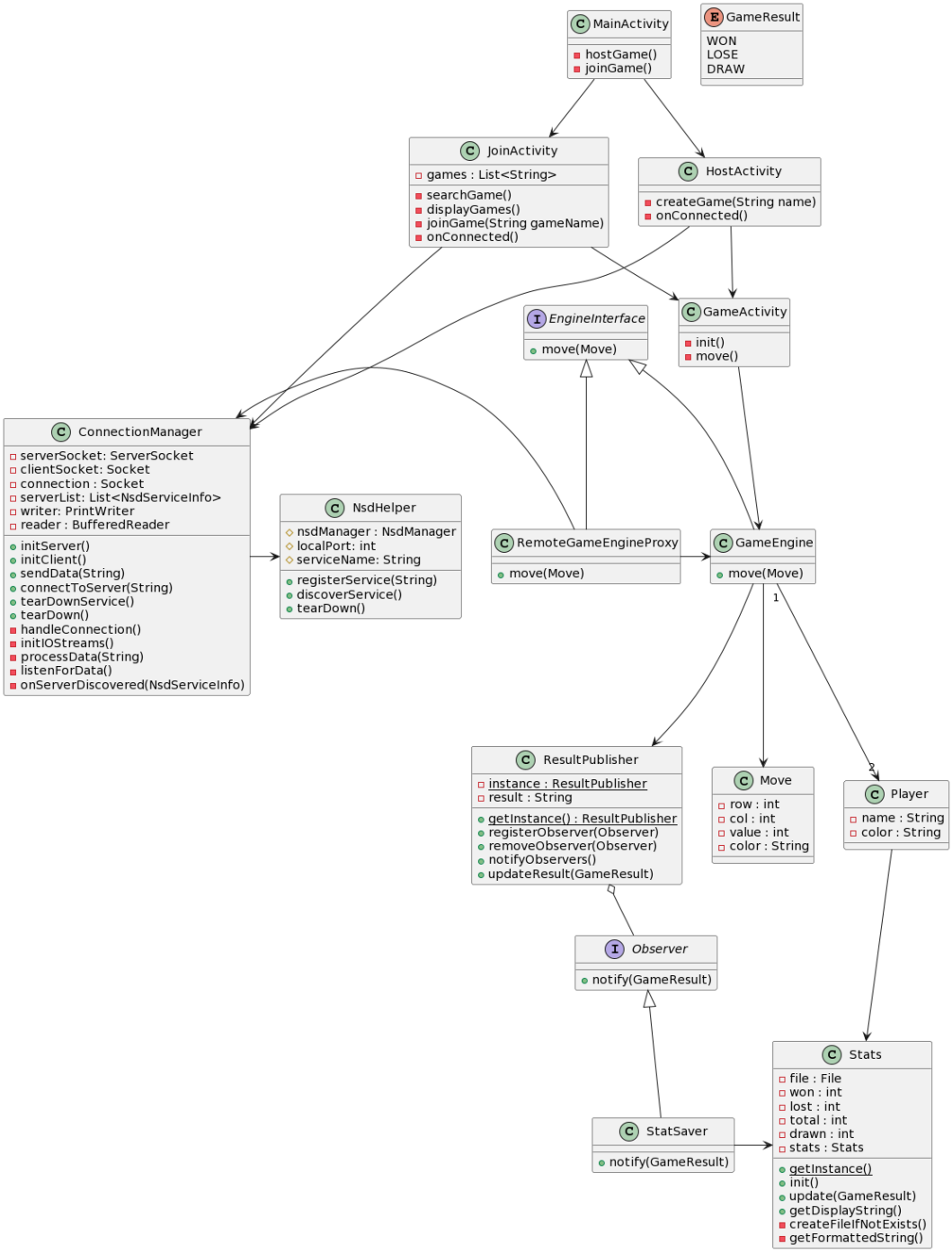
Work Done
We have worked on setting up the base for communication between two Android devices running our application. We used Network Service Discovery (NSD) API to discover game hosting users on the local network. Once that was successfully completed, we worked on creating a ConnectionManager to handle all communication requests between the client and server. After setting up the client and server sockets, a basic implementation of the proxy pattern was implemented.
Parallely, we worked on creating UI pages for the Home, Join, and Host screens, and then wired the pages to the ConnectionManager and the proxy.
Finally, we have almost completed the observer pattern to update the game result. We also created the data classes required for storing the game stats and player info.
Breakdown

- Varad worked on the networking part including NSD, ConnectionManager, and Proxy pattern.
- Aishwarya worked on the UI, Observer pattern, and the data classes

Changes and Issues
In terms of the overall design, nothing has changed much. The class diagram is basically the same as before, except the addition of some helper classes and some methods and variables. Setting up communication between two devices was more challenging than expected. Which is why we added a separate NsdHelper class and changed the structure of the ConnectionManager class.

Patterns
As mentioned above, we have implemented the Proxy and Observer patterns along with the Singleton pattern. The Proxy pattern has helped in abstracting the underlying network connection. Moving forward, it will make syncing the two game engines (on the two separate devices) much easier.

Class Diagram



1. The Stats and ResultPublisher classes demonstrate the **Singleton** pattern.
2. The **Observer** pattern is implemented through the ResultPublisher, Observer, and StatSaver classes together.
3. The EngineInterface, GameEngine, and RemoteGameEngineProxy classes demonstrate the **Proxy** pattern. RemoteGameEngineProxy uses ConnectionManager to send and receive game data (like moves) between the two devices.

Plan for Next Iteration

For the final implementation, the State pattern is yet to be implemented. We also have to implement the game board UI using Android Canvas. We will be working with the raw Canvas API. Thus, setting up a grid pattern and mapping the grid squares to correct screen coordinates will be a major challenge. Next task would be to identify user touch on the board when a move is made. The board will be updated with the correct colored stone and at the same time, the RemoteGameEngineProxy will be used to notify the peer player about the move. Next to the game board, we also plan to display game stats for both the players.
By 12/7, we plan to present a fully functional application of Gomoku. Two players on different Android phones will be able to play over the local network using our application. They will be notified when the game ends, along with the result of the game.