Networking Considerations for Hooop!

Our current Hooop! design operates entirely on a single computer, with GameBoard, frogs, bridges, and action cards all stored in local memory. To enable online multiplayer, the main modifications involve how game information is shared between devices and how turn control is managed.

Passing Information:
Currently, the full game state is stored in the GameSystem and Gameboard objects. These structures are already clean and modular, so that they can be serialized and sent across computers as small update messages. This makes it practical to synchronize the board without transmitting the entire state every time.

Control Across Computers:
Locally, the game controls turn using the GameSystem. For networking, one computer should act as the host that maintains the official version of the game. Other players send their moves to the host, who validates them, updates the board, and broadcasts the results. This prevents conflicts and keeps all players synchronized.

Setup Changes:
Instead of immediately starting a local game, we would need a setup screen with options like "Join Game." The host would wait in a lobby until all players are connected. Once everyone is ready, the host starts the game and sends the initial game state to all clients.

User Interface:
Most of the existing GUIs could stay the same during gameplay. Only the startup flow needs additions (lobby, join option). All players would use the same board interface once the match begins.

Game State Handling:
Because the model classes already hold all information cleanly, it would be easy to take the needed parts of the state and transmit them. We only need to send enough data for each computer to update its own local copy rather than sending everything.

Turn Control and Game Flow:
The host would determine whose turn it is and send "Next Player" messages. On a client machine, the UI would unlock only when it is that player's turn. Similarly, starting the game, ending the game, and detecting a win would all happen on the host and be communicated to all others.

Overall, the current object-oriented design already supports networking well. Only the control flow, setup screens and communication layer would need to be added, not a redesign of the core game logic