UML Class Diagram
Network

<<interface>>
AppServer

connect(): Server

<<interface>>
Client

updateGamesList(List<GameDetailsView>: void

showError(String, boolean): void

updatePlayersList(List<String>): void

gameHasStarted(): void

modelChanged(GameView): void

gameFinished(GameView): void

ping(): void

<<interface>>
Server

register(Client): void

addPlayerToGame(int. String): void

create(int, int, String): void

getGamesList(): void

performTurn(int, Point...): void

ping(): void

AppServerImpl

- instance: AppServerImpl
- executorService: ExecutorService
- serverIP: String
- socketPort: int

+ main(String[]): void
+ getInstance(): AppServerImpl
- startRMI(): void
- startSocket(): void

AppClient

- uiType: UIType
- connectionType: ConnectionType
- ip: String
- port: int

+ main(String[]): void
- askArgs(Scanner): void
- parseArgs(String[]): void
- startRMI(): void
- startSocket(): void

ServerImpl

- model: Game
- controller: Controller
- client: Client
- playerIndex: int
- pingpongThread: PingPongThread

ServerExecutor

- server: Server
- client: Client
- executorService: ExecutorService

ServerStub

- ip: String
- port: int
- oos: ObjectOutputStream
- ois: ObjectInputStream
- socket: Socket

+ receive(Client): void
+ close(): void

ClientImpl

- server: Server
- startUI: StartUI
- gameUI: GameUI
- pingReceived: boolean

+ initialize(): void
+ run(): void
- closeConnection(): void

ClientSkeleton

- oos: ObjectOutputStream
- ois: ObjectInputStream

+ receive(Server): void

PingPongThread

- pong: boolean
- server: ServerImpl

+ run(): void
- handlePlayerDisconnection(): void
+ pongReceived(): void

<<enumeration>>
Methods

UPDATE_GAMES_LIST

UPDATE_PLAYERS_LIST

SHOW_ERROR

GAME_HAS_STARTED

MODEL_CHANGED

GAME_FINISHED

PING

<<enumeration>>
Methods

JOIN

CREATE

GET_GAMES_LIST

PERFORM_TURN

PONG