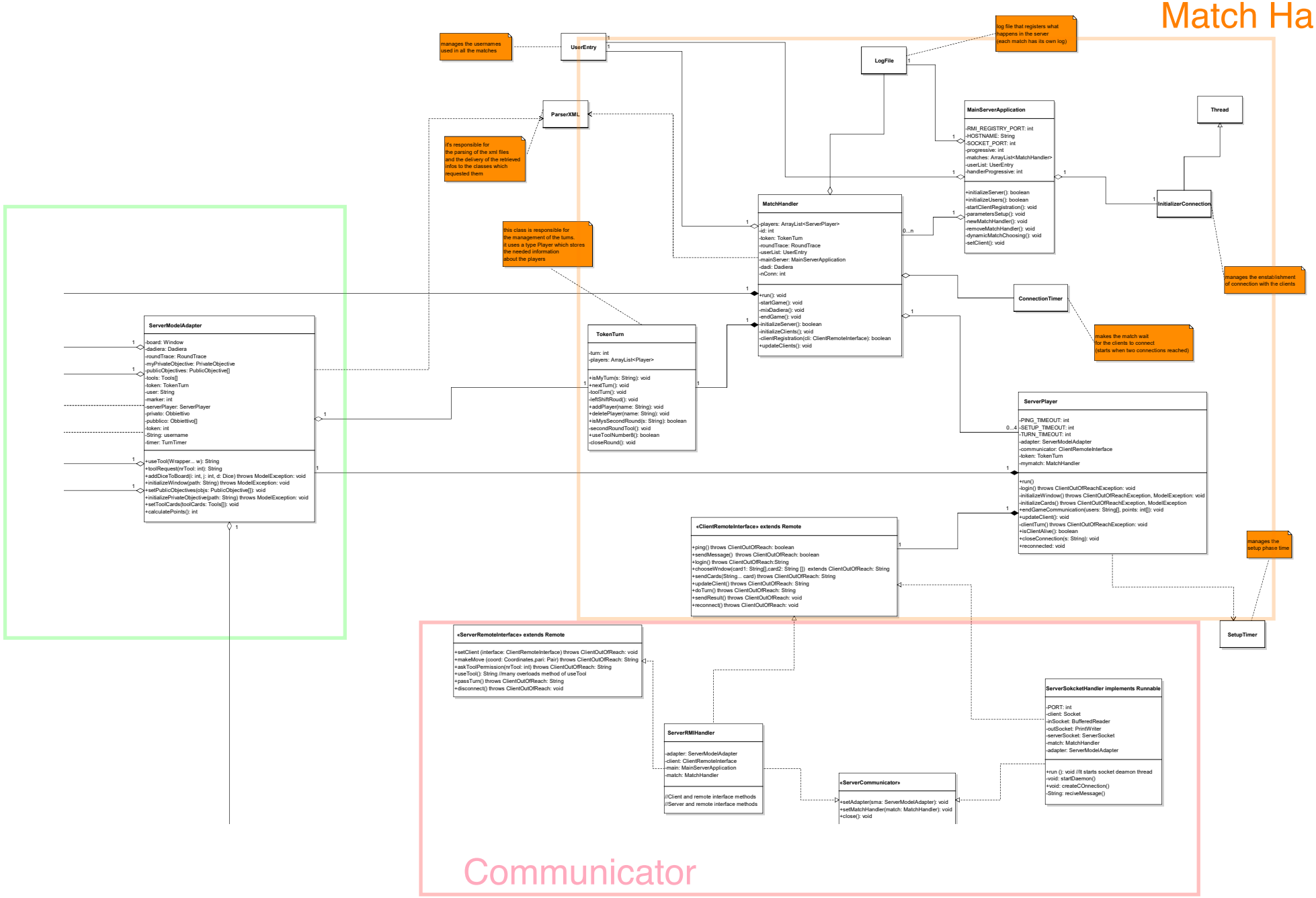
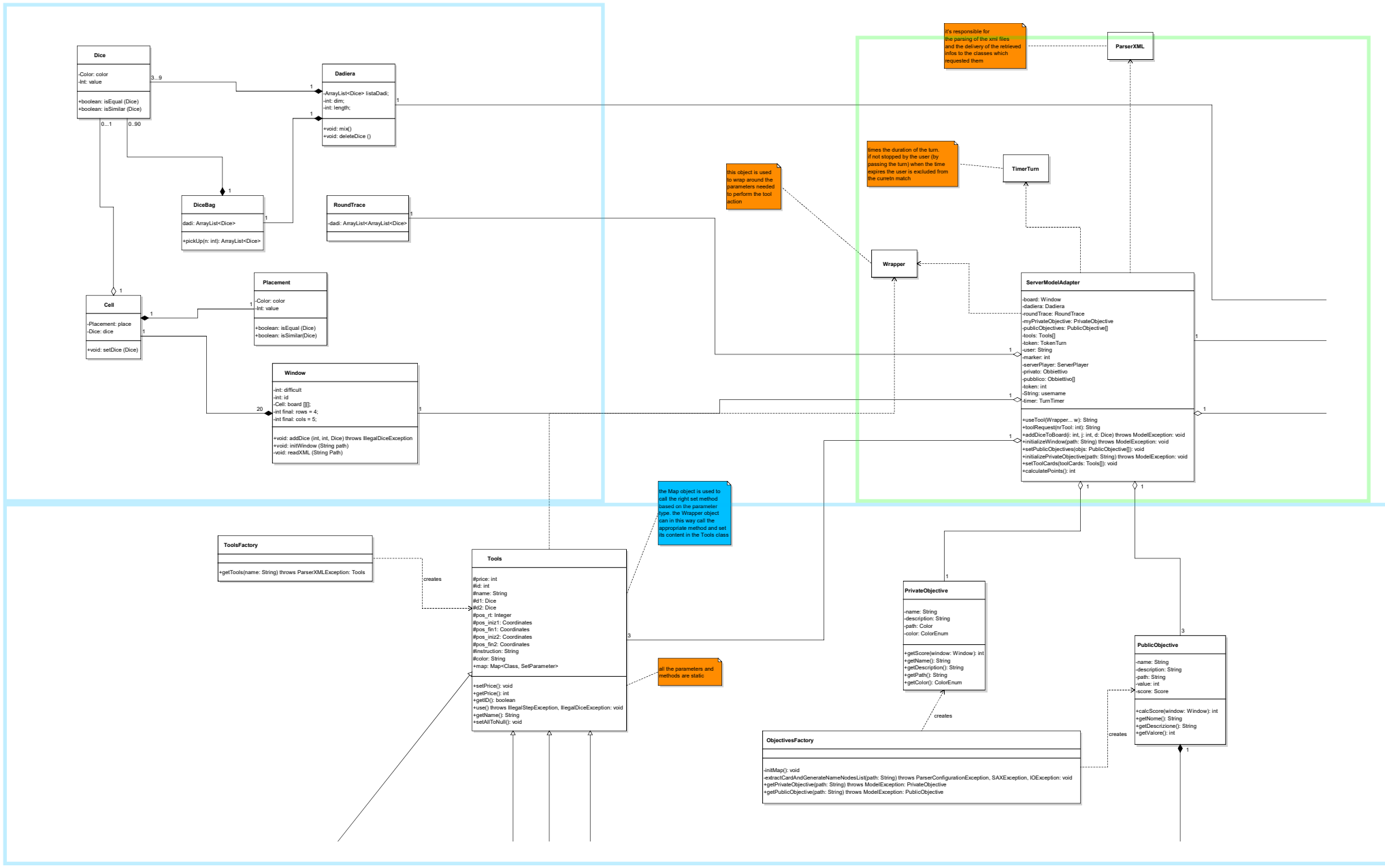
Server

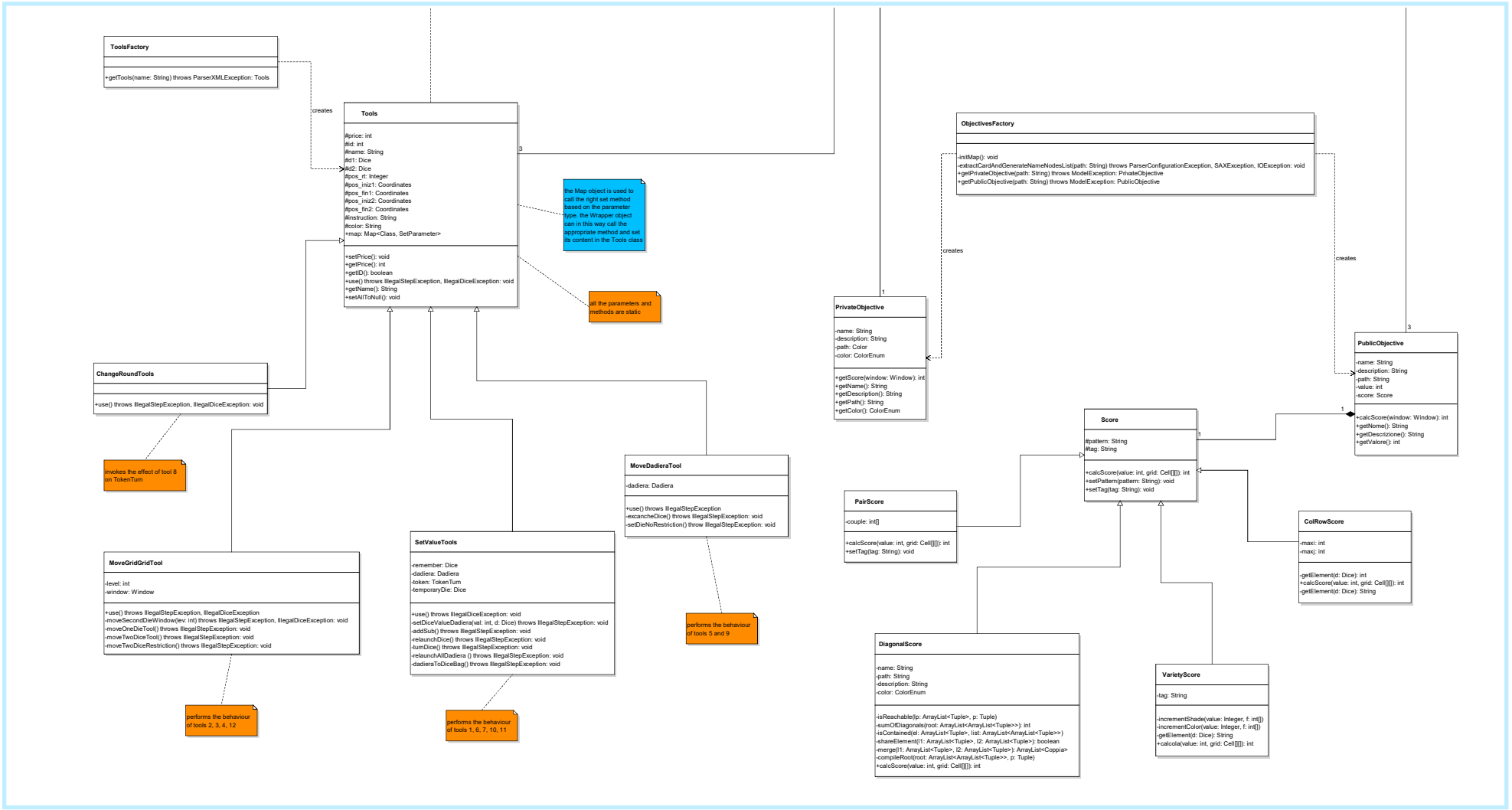

Match Handling
manages the usernames used in all the matches
UserEntry
log file that registers what happens in the server (each match has its own log)
LogFile
ParserXML
it's responsible for the parsing of the xml files and the delivery of the retrieved infos to the classes which requested them
MainServerApplication
-RMI_REGISTRY_PORT: int
-HOSTNAME: String
-SOCKET_PORT: int
-progressive: int
-matches: ArrayList<MatchHandler>
-userList: UserEntry
-handlerProgressive: int
+initializeServer(): boolean
+initializeUsers(): boolean
-startClientRegistration(): void
-parametersSetup(): void
-newMatchHandler(): void
-removeMatchHandler(): void
-dynamicMatchChoosing(): void
-selClient(): void
Thread
InitializerConnection
manages the enstablishment of connection with the clients
this class is responsible for the management of the tums. it uses a type Player which stores the needed information about the players
MatchHandler
players: ArrayList<ServerPlayer>
-id: int
-token: TokenTurn
-roundTrace: RoundTrace
-userList: UserEntry
-mainServer: MainServerApplication
-dadi: Dadiera
-nConn: int
+run(): void
-startGame(): void
-mixDadiera(): void
-endGame(): void
-initializeServer(): boolean
-initializeClients(): void
-clientRegistration(cli: ClientRemoteInterface): boolean
+updateClients(): void
ConnectionTimer
makes the match wait for the clients to connect (starts when two connections reached)
ServerModelAdapter
-board: Window
-dadiera: Dadiera
-roundTrace: RoundTrace
-myPrivateObjective: PrivateObjective
-publicObjectives: PublicObjective[]
-tools: Tools[]
-token: TokenTurn
-user: String
-marker: int
-serverPlayer: ServerPlayer
-privato: Obbiettivo
-pubblico: Obbiettivo[]
-token: int
-String: username
-timer: TurnTimer
+useTool(Wrapper... w): String
+toolRequest(nrTool: int): String
+addDiceToBoard(i: int, j: int, d: Dice) throws ModelException: void
+initializeWindow(path: String) throws ModelException: void
+setPublicObjectives(objs: PublicObjective[]): void
+initializePrivateObjective(path: String) throws ModelException: void
+setToolCards(toolCards: Tools[]): void
+calculatePoints(): int
TokenTurn
-turn: int
-players: ArrayList<Player>
+isMyTurn(s: String): void
+nextTurn(): void
-tooTurn(): void
-leftShiftRoud(): void
+addPlayer(name: String): void
+deletePlayer(name: String): void
+isMysecondRound(s: String): boolean
-secondRoundTool(): void
+useToolNumber8(): boolean
-closeRound(): void
ServerPlayer
-PING_TIMEOUT: int
-SETUP_TIMEOUT: int
-TURN_TIMEOUT: int
-adapter: ServerModelAdapter
-communicator: ClientRemoteInterface
-token: TokenTurn
-mymatch: MatchHandler
+run()
-login() throws ClientOutOfReachException: void
-initializeWindow() throws ClientOutOfReachException, ModelException: void
-initializeCards() throws ClientOutOfReachException, ModelException
+endGameCommunication(users: String[], points: int[]): void
+updateClient(): void
-clientTurn() throws ClientOutOfReachException: void
+isClientAlive(): boolean
+closeConnection(s: String): void
+reconnected: void
«ClientRemoteInterface» extends Remote
+ping() throws ClientOutOfReach: boolean
+sendMessage() throws ClientOutOfReach: boolean
+login() throws ClientOutOfReach:String
+chooseWndow(card1: String[],card2: String []) extends ClientOutOfReach: String
+sendCards(String... card) throws ClientOutOfReach: String
+updateClient() throws ClientOutOfReach: String
+doTurn() throws ClientOutOfReach: String
+sendResult() throws ClientOutOfReach: void
+reconnect() throws ClientOutOfReach: void
manages the setup phase time
SetupTimer
«ServerRemoteInterface» extends Remote
+setClient (interface: ClientRemoteInterface) throws ClientOutOfReach: void
+makeMove (coord: Coordinates,pari: Pair) throws ClientOutOfReach: String
+askToolPermission(nrTool: int) throws ClientOutOfReach: String
+useTool(): String //many overloads method of useTool
+passTurn() throws ClientOutOfReach: String
+disconnect() throws ClientOutOfReach: void
ServerRMIHandler
-adapter: ServerModelAdapter
-client: ClientRemoteInterface
-main: MainServerApplication
-match: MatchHandler
//Client and remote interface methods
//Server and remote interface methods
«ServerCommunicator»
+setAdapter(sma: ServerModelAdapter): void
+setMatchHandler(match: MatchHandler): void
+close(): void
ServerSokcketHandler implements Runnable
-PORT: int
-client: Socket
-inSocket: BufferedReader
-outSocket: PrintWriter
-serverSocket: ServerSocket
-match: MatchHandler
-adapter: ServerModelAdapter
+run (): void //It starts socket deamon thread
-void: startDaemon()
+void: createCOnnection()
-String: reciveMessage()
Communicator

Model

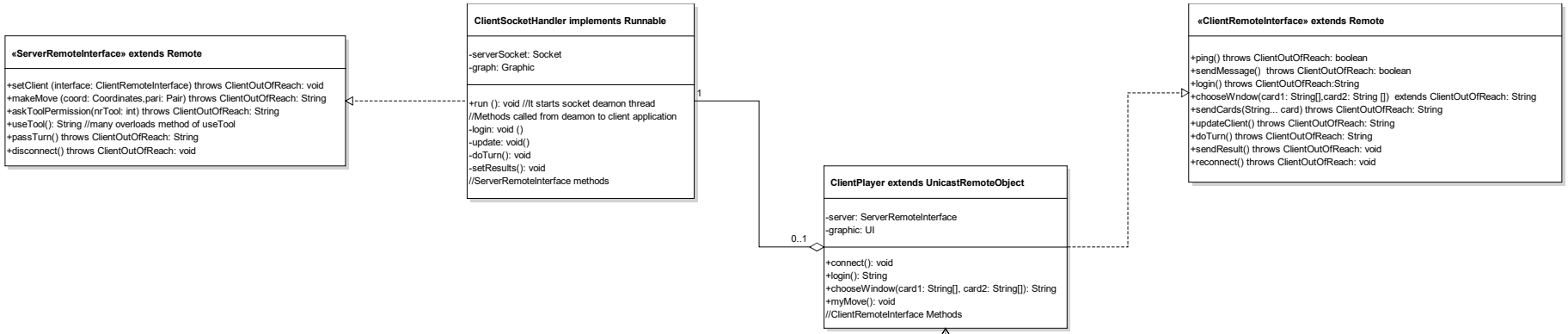


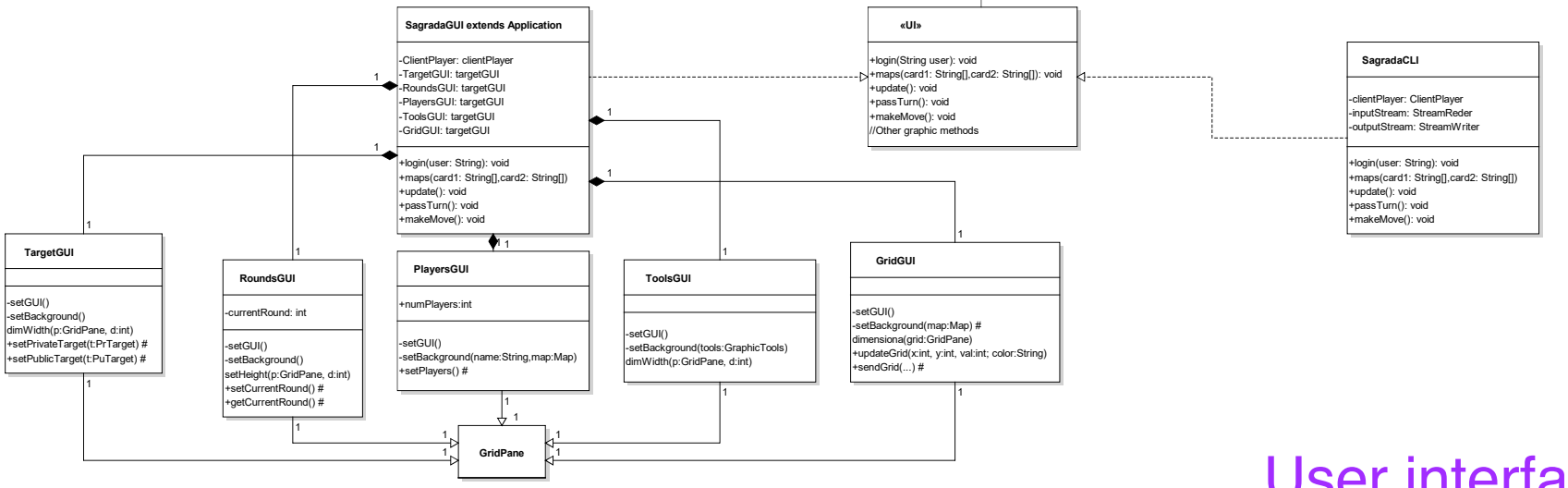
Model continue

Model continue



Communicator





User interface

Client