type User is TupleType {playerId:int, username:String }   
type Account is TupleType { username:String, password:String}  
type Player is enum {Player1, Player2}  
type Command is enum {MoveAt, FireAt, DropMine, PickupMine, RotateBoat}   
type CommandResult is enum {Success, Collision, Hit, Miss}  
type Tile is TupleType {x:int, y:int}  
type Cell is TupleType{ aTile:Tile, currentHealth:int, maxHealth:int}  
type Boat is TupleType { enum {Radar, …}, cells:Sequence(Cell)}  
type Base is TupleType {cells:Sequence(Cell)}  
type Reef is TupleType { tiles:Sequence(Tile)}  
type InGamePlayer is TupleType {p:Player, u:User, boats:Sequence(Boat), base:Base}  
type Game is TupleType{player1:InGamePlayer, player2:InGamePlayer, reef:Reef, mines:Sequence(Tile)}  
type GameSettings is TupleType {volume:int, fullscreen:boolean}

**Message (Type) Declarations:**

login(acc:Account)  
register(new\_acc: Account)  
newGame(roomId:int)  
loadGame(gameId:int)  
joinRoom(roomId:int, user:User)   
createRoom(user:User)   
reefDecision(decision:Boolean)  
swapBoats(boat1:Boat, boat2:Boat)   
spectateGame(gameId:int)  
sendMessage(message:String)  
boatCommand(command:Command, tile:Tile)   
quit()  
yourTurn(enemyCommand:Command, modified\_cells:Sequence(Cell))  
boatCommandResult(result:CommandResult, atTile:Tile, modified\_cells:Sequence(Cell))   
saveGameAnswer(answer:Boolean)  
spectatedGame(initialGameState:Game, commands:Sequence(Command), initialSettings:GameSettings)  
spectatedGameEvent(result:CommandResult, atTile:Tile, modified\_cells:Sequence(Cell))  
initialGame(init:Game, initialSettings:GameSettings)  
loadedGame(load:Game, initialSettings:GameSettings)   
reefConfiguration(conf:Reef)   
room(roomId:int, load\_game\_Ids:Sequence(int))   
receiveMessage(message:String)  
allRooms(roomIds:Sequence(int))  
endgame(winner:Player)  
enemySwapBoats(boat1:Boat, boat2:Boat)   
registerResult(registered:Boolean)  
loginResult(loggedIn:Boolean)  
ping()  
pingResponse()  
playerDisconnected()