Operation: User :: logIn (username: String, password: String)  
Scope: Account; Game  
Messages: Client :: { sendLoginInfo(), userValid(), userInvalid() }; Server :: { validateUser(username:String, password:String) }  
Pre:  
Post:  
Description:

Operation: signUp  
Scope: Account; Game  
Messages: Client :: { newUserCredentialsValid(),newUserCredentialsInvalid() }; Server :: { validateNewUserCredentials(username:string, password:string) }  
Pre:  
Post:  
Description:

Operation: reviewStatistics  
Scope: Account; Statistics  
Messages: Client :: { requestStatistics(player:String); provideStatistics(stats:Statistics) }; Server :: { requestStatistics(player:String) }  
Pre:  
Post:  
Description:

Operation: matchUp  
Scope: Game  
Messages: Client :: { requestPlayers(); challengePlayer(username:String); acceptGame(from:String), rejectGame(from:String) }; Server :: { requestAllCurrentPlayers(); requestMatching(from:String, to: String); playerAcceptsGame(from:String, to:String); playerRejectsGame(from:String, to:String) }; Client :: { allCurrentPlayers(players:List); askIfPlayerWantsToJoin(to: String); gameRejected(to:String); gameAccepted(to:String) }  
Pre:  
Post:  
Description:

Operation: setupGame  
Scope: GameConditions; Game  
Messages: Client :: { sendNewConditions(conditions:GameConditions); sendConfirmationOfConditions(); newGameConditions(conditions:GameConditions); startGame(conditions:Conditions) }; Server :: { submitGameConditions(conditions:GameConditions); submitAgreement() }; Screen :: { displayGameConditions(conditions:GameConditions); acceptGameConditions(); displayNewGame(conditions:Conditions) }  
Pre:  
Post:  
Description:

Operation: loadGame  
Scope: Game;   
Messages: Client :: { loadGame(name); loadGameSuccess(); loadGameFailure\_e() }; Server :: { loadGameState(name:String); }  
Pre:  
Post:  
Description:

Operation: saveGame  
Scope: Game;   
Messages: Client :: { saveGame(name:String); saveGameSuccess(); saveGameFailure\_e() }; Server :: { saveGameState(name:String); }; Screen :: { saveGameSuccessMessage();saveGameFailureMessage(); listLoadableGames(names:List); loadGameSuccessMessage(); loadGameFailureMessage() }  
Pre:  
Post:  
Description:

Operation: selectShips  
Scope: Game; ShipState; Ship  
Messages: Client :: { listSelectableShips(ships:List); shipSelectionValid(); shipSelectionInvalid(); select(ship:Ship); deselect(ship:Ship); submitShipSelections() }; Server :: { validateShipSelection(ships:List) }; Screen :: { displayShipSelectionScreen(ships:List) }  
Pre:  
Post:  
Description:

Operation: placeShips  
Scope: Game; ShipState; Ship  
Messages: Client :: { placeShip(ship:Ship); removeShip(ship:Ship); submitShipPlacement();placementValid(); placementInvalid\_e(); } ; Server :: { alertOfShipPlacement(shipPlacements:List) } ; Screen :: { displayPlaceShipsPage(); displayInvalidPlacementPage(); displayValidPlacementPage() }  
Pre:  
Post:  
Description:

Operation: makeAction  
Scope: Game  
Messages: Client :: { makeAction(action:InGameAction); actionValid(); actionInvalid\_e(); playerWin(); playerLose() }; Server :: { submitAction(action:InGameAction) }; Screen :: { displayActionInvalidPage() }  
Pre:  
Post:  
Description:

Operation: updateTurn  
Scope: Game  
Messages: Screen :: { displayNewGameScreen() }; Client :: { newGameScreen(shipPlacement:List, shipDamage:List) }  
Pre:  
Post:  
Description: