Class/method	Functionality	Notes
Main.cs	**Component of: Main Object (MO)**	
Awake()	add Message Queue (MsgQ) as component	classes can be added as components to game objects. var "gameObject" refers to current GO; presumably causes calls to Awake(), Start(), Update(), ... to be initiated in those classes.
Awake()	add Connection Manager (ConnMgr) as component	
Awake()	init class NetworkRequestTable	**external call: NetworkReqTable.init()**; creates dictionary of net requests using generic class "Type" to define references to classes RequestLogin, RequestHeartbeat
Awake()	init class NetworkResponseTable	**external call: NetworkResponseTable.init()**; creates dictionary of net responses using generic class "Type" to define references to class ResponseLogin
Awake()	init class SpeciesTable	**external call: SpeciesTable.initialize()** loads SQLight DB WoB_DB.db
Start()	Load "Login" scene	scene = "level" in scripts. Presumably this results in processing of **login.cs.** Note: loading levels destroys existing GOs unless DontDestroyOnLoad has been called (in Awake().
Start()	declare var for ConnMgr; call method ConnMgr.setupSocket()	**external call: ConnMgr.setupSocket()**
Start()	Coroutine call: RequestHeartBeat()	
RequestHeartBeat()	create instance of class RequestHeartBeat (RequestHB), a subclass of NetworkRequest.	**external call: RequestHB.send()**. RequestHB.send() returns instance of a (network) request containing GamePacketStream buffer with id type CMSG_HEARTBEAT.
RequestHeartBeat()	forward request to ConnMgr	**external call: ConnMgr.send(request)**
RequestHeartBeat()	recursive call to this Coroutine	
ConnMgr.cs	**instantiated as component of MO by main.cs**	look to c# documentation for more information about TCPclient, MemoryStream, ...
Update()	call readSocket()	
setupSocket()	create TCP client instance and open network stream (theStream)	??Where are constants REMOTE_HOST and REMOTE_PORT defined?? Constants.cs... called by??
readSocket()	process stream envelope: first two bytes indicates length of stream=>dataStream; next short int (read from dataStream) indicates response type=>response_id.	
readSocket()	create instance of class specified by response_id and attach dataStream to the instance, unknown [response-class]	**external call: NetworkResponseTable.get()**; creates instance of class indicated by response_id, e.g. ResponseLogin
readSocket()	extract data from dataStream	**external call: [response-class].parse()**; parses the data (e.g. for ResponseLogin, reads and stores status, user_id, username and last_logout)
readSocket()	store parsed data into var "args" (subclass of ExtendedEventArgs class specific to - and stored in same file as - [response-class])	**external call: [response-class].process()**
readSocket()	add arguments to MsgQ (MO component)	**external call: MsgQ.AddMessage()**
send()	request is copied to the open network stream (theStream), where it is forwarded to the server	send() is called by external methods making network requests, such as **main.RequestHB()** and **login.Submit()**
Login.cs	**Presumably "activated" by loading of "Login" scene in main.cs**	
Awake()	configure message event callbacks	**external call: MsgQ.AddCallback()**; e.g. for event id SMSG_AUTH, set callback to Login.ResponseLogin()
OnGUI()	draw login window with call to MakeWindow() and Submit()	
Submit()	process user input. Call local method RequestLogin() to package request and then forwards request to ConnMgr	**external call: ConnMgr.send()**
RequestLogin()	create instance of RequestLogin with user login information	**external call: RequestLogin.send()**; returns instance of a (network) request containing GamePacketStream buffer with id type CMSG_AUTH (i.e. login)
ResponseLogin()	set Constants.USER_ID to user_id in args	
MsgQ.cs	**instantiated as component of MO by main.cs**	
Awake()	create empty callbackList dictionary and msgQueue queue of ExtendedEventArgs	look at c# documentation for more information about callbacks
AddCallBack()	add specified callback method to list	called from login.Awake() to configure ResponseLogin (and possibly other classes to configure other Responses)
Update()	if there is data in msgQueue (see ConnMgr->readSocket()), make "callback" to function indicated by event_id with msg args	**external call: Login.[callback function]**; e.g. ResponseLogin() contained in login.cs
AddMessage()	add specified EventArgs to msgQ	called by ConnMgr.readSocket()