Note: Our program for iteration 2 was built with the Go language and therefore does not implement classes. The objects used to represent our code in this UML sequence diagram are the packages we created that are responsible for communicating with the registry and peer handling. Neightbour Peer pkg.peerProc main.go pkg.registry Registry Process Run Peer Process Request Registry Address Send Registry Address Request Peer Address Send Peer Address registry. In it Registry Communicator ()sock.InitializeTcpClient() Send close connection message return registry.InitRegistryCommunicator() peer.InitPeerProcess(peerAddr, ctx) readSnip() return snip message sendSnip() sendPeerList() handleMessage() if msg == "snip" sends "snip" msg if msg == "peer" send "peer" msg if msg == "stop" send "stop" msg registry.InitRegistryCommunicator() sock.InitializeTcpClient() Send close TCP connection message return registry.InitRegistryCommunicator()