15-440: Lab 2

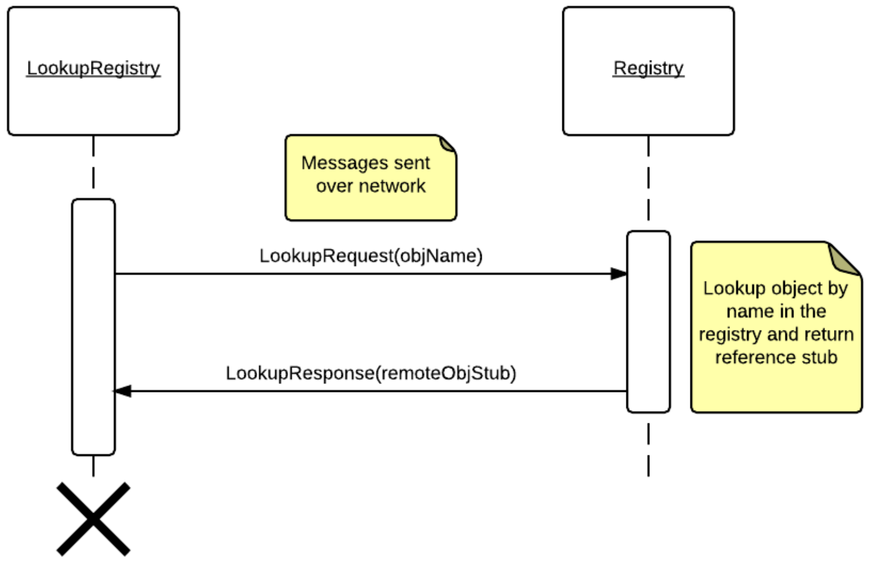
Spencer Barton (sebarton)
Emma Binns (ebinns)

September 11, 2014

1 Design

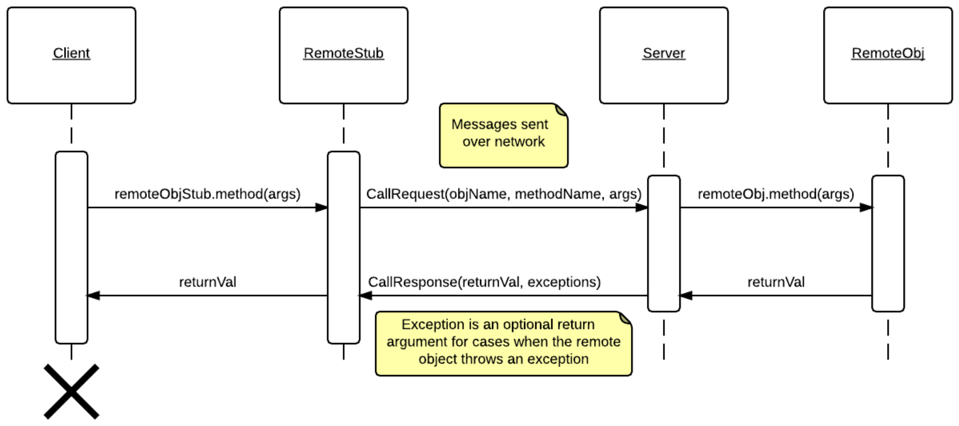
- UML diagram (Emma) - Why no skeleton

Registry Lookup


LookupRegistry
Registry
Messages sent over network
LookupRequest(objName)
Lookup object by name in the registry and return reference stub
LookupResponse(remoteObjStub)

In order to lookup a remote object the client uses their `LookupRegistry` to lookup an object by name string. The `LookupRegistry` creates a `LookupRequest` with the object name. This is sent to the server which looks up the object by name in its registry. If the object exists, a `RemoteObjectStub` is returned through a `LookupResponse` else the `LookupResponse` comes back with an exception.

Remote Method Invocation


Client
RemoteStub
Server
RemoteObj
Messages sent over network
remoteObjStub.method(args)
CallRequest(objName, methodName, args)
remoteObj.method(args)
returnVal
CallResponse(returnVal, exceptions)
returnVal
Exception is an optional return argument for cases when the remote object throws an exception

2 Implementation State

We completed the basic requirements.

The extras that were not added include a skeleton, compiler and loading class files. We chose not to implement skeletons as we were able to add that the necessary methods within `RemoteObject`.

In order to add a compiler the design would not need to change. When performing a registry lookup a remote object stub is returned. Currently this remote object stub must be instantiated earlier by the server. With a compiler this would simply be replaced by a call to the compiler during an object lookup.

Finally support for loading class files would require a little more work since another server socket would need to be implemented on the host the .class files. Beyond that the remote object stub would handle a .class request and another type of message could be implemented to handle this communication.

3 Running the Project

- Instructions (example run) - CHECK works Andrew machines (Emma)

4 Dependencies

There are no dependencies.

5 Testing

Our test in built into `PartyClient`. It automatically runs when `PartyClient` is run. The test runs each of the methods in the `Person` object for both a remote and local implementation. `Person.samePerson` is run using both remote and local objects as arguments to check that the server-side remote object implementation can handle either object type as an argument.