Project 5

CMSC 433
Programming Language Technologies and Paradigms
Fall 2002

Due Friday, December 13, 2002 at 6pm

Introduction
The goals of this project are to introduce you to distributed programming and teach you a bit about Java’s approach to distributed programming called RMI.

1 A Remote Log with RMI

In Project 1, a RemoteLogClient basically acts as a proxy for a Log running in some other location. To access and modify the remote Log, you relied on a custom communication protocol known to your RemoteLogClient and your RemoteLogServer. That is, your RemoteLogClient sent a specially coded message to your RemoteLogServer. Upon receiving the message, your RemoteLogServer decoded it, and then executed a local method based on the contents of the message. For example, each time you invoked the add method on your RemoteLogClient object, a message would be sent across the network to the RemoteLogServer and the add operation would be applied to the RemoteLogServer’s private Log. In essence, you were doing a remote method invocation.

Using custom protocols, however, is complicated and error-prone. Java RMI allows us to eliminate the need for custom protocols. It does this by making remote objects accesses look the same as local object accesses (It doesn’t really succeed, though ...).

To demonstrate this, you will modify your Project 1 code to use RMI, as follows:

1. Set up LocalLogs to be potentially-remote, by modifying the Log interface and the LocalLog implementation to extend from the appropriate RMI classes/interfaces.

2. Modify the LogRecord class appropriately. In particular, you must decide whether or not to make LogRecords remote objects or whether they should always be copied when passed as method arguments.
3. Copy your RemoteLogServer class to a new version called RMIRemoteLogServer. This class will do two things:

(a) Create a LocalLog instance and store it in a private field. You can make the Log have size 100.
(b) Register this LocalLog in the local rmiregistry with the name RemoteLog.

Notice that you now have substantially less code than your original RemoteLogServer!

4. Copy your RemoteLogClient class to a new version called RMIRemoteLogClient, having the following signature. NOTE: Some things have changed from the original version. The log object is of type Log, rather than LocalLog and the methods throw RemoteException.

```java
public class RMIRemoteLogClient implements Log {
    private Log log;
    public RMIRemoteLogClient (String remoteLogName) throws RemoteException;
    public void add (LogRecord l) throws RemoteException;
    public LogRecord[] getAll(long time) throws RemoteException;
    public void setFilter (String s) throws RemoteException;
}
```

Calling the constructor will look up the Log object with the given `remoteLogName` in the local RMI registry, and store it in the private log field. All of the remaining methods will simply forward calls to this private log object.

5. Change the HeapSortAlgorithm class’s main method to use the RMIRemoteLogClient object, rather than LocalLog or RemoteLogClient, created with name RemoteLog.

2 Running your Code

You can now test your new implementation by

- starting up an rmiregistry on port x33yy – where x is your section number and yy is your account number,
- starting up the RemoteLogServer using command line property -Dport=x33yy defined as before, and
- starting up the HeapSortAlgorithm using command line property -Dport=x33yy defined as before, and -Dhost=registryHost.
3 Other Notes

- The output should be the same as the output described in project 1.
- Your project will tested with code running across multiple machines. You should put the following files in the client code directory:
  - TBD
  - and, the following code in your server code directory:
    - TBD
- Don’t forget to compile the relevant classes with rmic (in addition to javac).
- We will go over an example from the java RMI tutorial in class. You should walk through this example yourself. If you can’t get the example to work, you probably won’t get your project to work either.
- You can use your wam account as the webserver for your downloadable code. Put the Server code and Client code in different places to ensure that your downloading the code properly.
- On Unix I put the java policy file in YourHomeDir/java.policy. There’s a sample policy file on the project 5 web page.
- I’ve never gotten the rmiregistry to work as shown in the RMI tutorial. Instead, I do the following:
  1. make a subdirectory with nothing in it,
  2. cd to that directory,
  3. unset the classpath variable (% unsetenv CLASSPATH), and then
  4. run the rmiregistry (% rmiregistry portNum &)
- More notes will definitely follow.