

Lec 19

Wed 3/28/01

// Server

main() variable name

{

StockImpl stock = new StockImpl();

new → Naming.rebind("StockServer", stock);

// Client

StockImpl myStock = Naming.lookup
new → ("StockServer");

myStock.get... can all be done
now with the myStock object.

"StockServer" is just a name to
pass to the client.

Server: define an interface containing
the list of methods that can be
called remotely.

① public interface Stock ~~implements~~
extends
Remote

{
public String getName (String symbol);
public double getStockValue
(String symbol);
}

// the purpose of this interface is
// to define what methods are
// accessible by the clients.

② public class StockImpl extends
UnicastRemoteObject implements
Stock

{
default constructor
public String getName (String symbol)
{
return "abc"; // just to test
} // if working

public double getStockValue
(String symbol)
{
return "12.7";
}

→ UnicastRemoteObject class provides function
to "export" the object!

exporting object: providing ServerSocket.
// all of this is hidden from you

Lec 19

wed 3/28/01

What if you don't want to attach
to the Unicast Remote Object?

UnicastRemoteObject.exportObject();
- a static function.

③ Create the object and register it.

main()

{

StockImpl st = new StockImpl();
Naming.rebind("StockServer", st);

}

Executing the Server:

- ① Create stub/skel classes.
→ make sure StockImpl is compiled.
- go to Dos prompt:

C:\CS256\rmistock

C:\CS256\> C:\JBuilder4\jdk1.3\bin\

rmic StockImpl rmistock StockImpl

rmic → remote method Invocation Compiler,
is a utility that creates two classes:
StockImpl_stub.class
StockImpl_skel.class

Using above will store new class files in rmistock directory / package.

In rmistock package, you should have

StockImpl.class
StockImpl-stub.class
StockImpl-skel.class

- ② Start Registry (do not start Registry from menu selection in JBuilder)

Do it in DOS:

c:\cs 256\> ~~JBuilder\jdk1.3\bin\rmiregistry~~
→ start, c:\JBuilder\jdk1.3\bin\rmiregistry

You can use the Doskey command so you only have to type this in once.

Note: you will see a new DOS window showing up.
rmiregistry is running in this window.
Don't close that window.

If the window closes by itself, rmiregistry did not run correctly.

rmiregistry doesn't print anything in its dos window to indicate that it is running.

Minimize the empty registry window
But don't close it.

Lec 19

Wed 3/28/01

How do I know rmiregistry is running?

- ① there will be the empty dos window in which rmiregistry runs.
- ② push ctrl - alt - delete only shows in the list, you will see rmiregistry listed.

(How to close a running rmiregistry?)

- ① Open the empty rmiregistry window.
press (ctrl - c).

or

- ② ctrl - alt - delete
end task → rmiregistry.

- ③ Running Server

start StockSimple from the file with the main method (like any other java application).

What do I do if I modify Stock or StockImpl or main on the server side?

- ① close registry
- ② compile all files.
- ③ use rmic to create stub, skel.
- ④ start rmiregistry.
- ⑤ execute Server

- this finishes the server side discussion

Lec 19

wed 3/28/01

Client Code:

Client should be written so it only knows about the interface (Stock)

Note: the object StockImpl is of two types: StockImpl, Stock

StockImpl st = new StockImpl()

→ st is of type StockImpl
and of type Stock.

Client side: maybe a JFrame

main()

Stock myStock = (Stock) Naming.lookup("StockServer");

► can only access methods in Stock.

public interface Stock extends Remote

public String setName (String symbol)
throws RemoteException
// must do this

Lec 19

Wed 3/28/01

tm
{

String str = myStock.getName("dvc");
}
catch (Exception ex)

{

// sends complete String object.



Serializable: can be translated into bits for sending across the network, so serialize & deserialize object for transport.

Assignment 7:

public interface Stock extends Remote
{

 public double getStockValue
 (*String symbol*)

 throws RemoteException;

 public StockAll getStockAll
 (*String symbol*)

 throws RemoteException;

}

Lec 19

Wed 3/28/01

public class StockAlt

{
String name;
double loValue;
double hiValue;

String setName()
{

{
double getLoValue()
{

}

}

Behind on boards:

Sockets

Servers

UnicastRemoteObject (export object
create