Outline

- Quiz #4
- Homework #3
- Distributed computing overview
- Programming RMI
Homework #3

- Good news
  - Ave. 13.8

- Bad news
  - Some people didn’t turn in
Distributed computing

- Sockets
  - Messages from A to B

- Distributed computing
  - A is doing some computation P
  - A asks B to compute some function F
Sockets
Distributed Computing

Server
Codebase
function F

Client
Application

?
Questions to ask

How does...
1. P know how to call F
2. P know that F resides on B
3. P send its message to B and B send the answer back
4. B handle F so it can be invoked

When does
5. The user/programmer of P have to know the answers to 1 and 2

How does
6. The system provide quality of service, security and management
Distributed comp. approaches

- (RPC)
- RMI
- DCOM
- CORBA/JIDL
- Web services
RMI

1. P and F are written in Java
   - F’s interface is available to P
2. Destination is an input
3. Standard part of Java API
4. RMI registry must be running on B
5. (a) compile-time (b) run-time
6. Not much
RMI
DCOM

1. P and F are implemented in COM
   - A type library for F is available
2. Destination encoded in registry
3. Part of Windows OS
4. Loads and runs autonomously
5. Standard answers
6. Some MS tools
DCOM

Server

COM object
public method

COM

Client

COM
Application

DCOM protocol
CORBA

1. An IDL description of F is available
2. The ORB knows
   - inter-ORB communication
3. Request sent to ORB
4. Destination ORB activates and runs
5. Standard answers
6. ORB vendors supply
CORBA
Web services

1. P gets XML description of F
2. Service registry knows
3. HTTP/SOAP interaction
4. web application server invokes F
5. Both at run time
6. Emerging
Web services
Trade-offs

- RMI
  - Java only
- DCOM
  - Windows only
- CORBA
  - Must by ORB software
- Web services
  - ??
Elements of an RMI application

- Remote object
  - contains the methods we want to call
- RMI server
  - a program that exposes the remote object
- RMI registry
  - built-in intermediary
- RMI client
  - program that accesses the remote object
Remote Object

- Must implement a remote interface
- UnicastRemoteObject convenience class
Remote Interface

- Remote object’s signature
- Must extend java.rmi.Remote
- All methods must throw RemoteException
RMI Registry

- Distributed with the JDK
- No application-specific arguments
RMI Server

- Program creates instance of Remote Objects
- Names and publishes them in the registry
- Must set security policy to allow remote connection
RMI Client

- Compiled using the Remote Interface
- Contacts remote registry
- Downloads stub for remote object
- Calls methods on the stub
Stub

- conforms to the remote interface, but
- connects to remote JVM
- marshals the parameters for F
- waits for the result of F,
- unmarshals the return value or exception returned, and
- returns the value to the caller.
Serialization

- Alternative to remote object
- Transmit the object itself
- Objects passed as parameters to remote methods
- implement java.io.Serializable
  - no code necessary
RMI Example

- WeatherInfo
  - serialized class
- IWeatherData
  - remote interface
- WeatherData
  - remote object
- WeatherServer
  - RMI server
- WeatherInfoClient
  - RMI client
Deployment

- Distribute
  - remote interfaces
  - definitions of serializable classes
- Run rmic to generate stubs for remote objects
- Run registry
- Run RMI server
CORBA

- not “COBRA”!
- Common Object Request Broker Architecture
- Language/platform-independent distributed object computing
Java IDL

- Similarities
  - Registry $\leftrightarrow$ ORB
  - remote interface

- Differences
  - CORBA is not Java-specific
  - Service brokering
Process

- Write IDL
- Run idltojava
  - Creates Java interface
  - Creates stubs
  - Creates skeletons
- Create remote object
  - Implements IDL interface