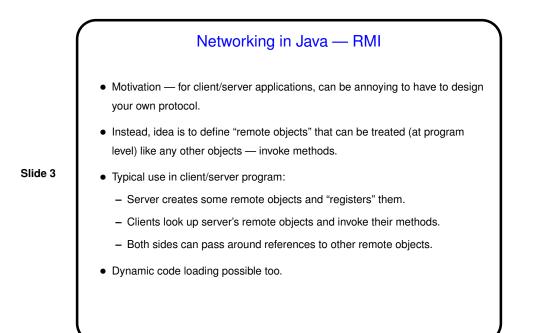
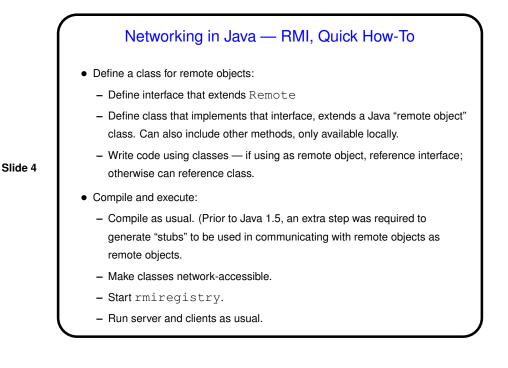


Slide 2





Networking in Java - RMI

- Example revised chat program. Design is somewhat more elaborate than absolutely necessary, in an attempt to be modular and flexible:
 - Common interface ChatParty for remote objects for both client and server, with subinterfaces ChatClient and ChatServer, and classes implementing all of these.

Slide 5

- Interface ChatClientUI for non-remote local UI for clients, with two implementations.
- Need for multithreading in server goes away all handled by RMI under the hood (though we still need to be careful about possible concurrent access to variables — experiment suggests RMI may use multiple threads). In client UI, however, we still need separate threads to get input from the user and listen for messages from the server.

Threads in Java, Revisited

- Earlier in the semester we talked a little bit about multithreading in Java.
 Basic functionality starting up new threads, coordinating actions of different threads has been part of Java from the beginning, but more because it's a nice model than for performance reasons.
- Slide 6
- With multicore machines becoming mainstream, though, using threads to improve performance is becoming more important. Much, much useful functionality in java.util.concurrent.

