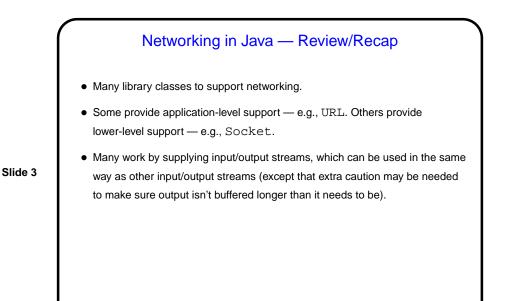


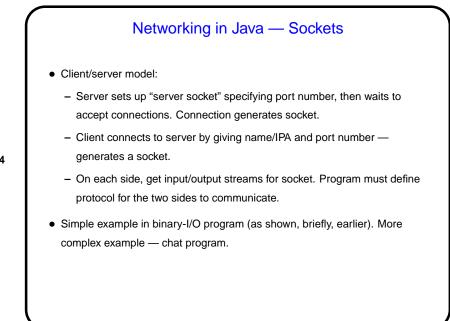
More Administrivia
"What about our grades?" You will get information by e-mail as soon as I have it.
Recall(?) weights from syllabus:

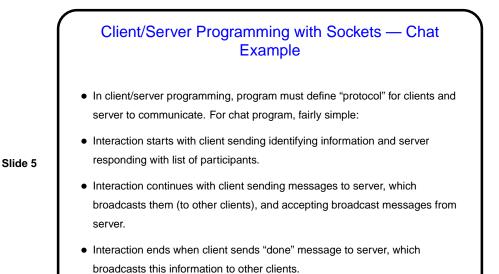
50 points class participation (attendance).
30 points quiz scores (lowest dropped).
300 points exams (100 midterm, 200 final).
480 points homework.

Late penalties will be reduced for most homeworks — say 5% per day with a maximum of 25%.

Slide 1

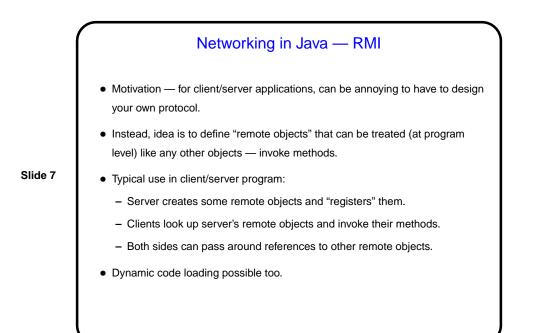


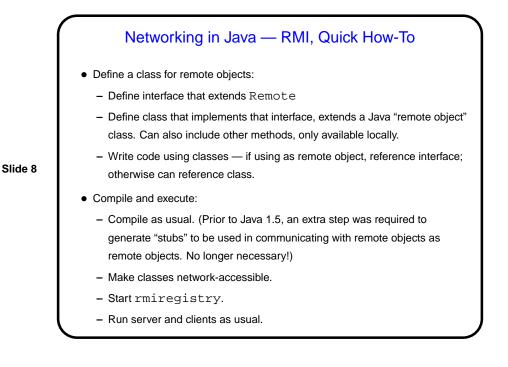




Client/Server Programming with Sockets - Chat Example, Continued

- Code is fairly simple classes for client and server, plus inner class for server to keep track of clients. Only tricky bits are related to concurrency ...
- · Server needs to be able to communicate with multiple clients asynchronously (i.e., no way to know which one will send a message next). One way to deal with this - start a new thread for each client. Must then be sure these threads don't concurrently modify shared data (here, list of clients).
- Client needs to be able to present GUI and also listen for messages broadcast by server. Less coding here since GUI runs in its own thread automagically, so we can use the main thread to listen for message from server. Only complication is that anything in this thread that needs to change the GUI must use SwingUtilities.invokeLater to be sure changes happen in event dispatch thread.





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 9

- 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.

Course Recap — What Did We Do?

- Java basics.
- Object-oriented programming polymorphism, inheritance, etc. Not stressed much in class, but game is a good example of a non-trivial o-o design.
- Basic ADTs stacks, queues, trees (sorted and heaps); different implementations (arrays versus dynamic data structures using references).
- Recursion review.
- Tour of the Java libraries GUIs, graphics, I/O; a very little about threads and networking.
- A fairly large programming project involving using someone else's code.
- To get a sense of what you learned compare what you knew in January to what you know now.

