Slide 1

Administrivia

- Homework 7 code due today. Accepted through Friday at noon without penalty. Grades on all homework will be e-mailed as available.
- Final December 17 at 8:30am. Homework 8 (design and code) due at that time. Not accepted past December 17 at 5pm.
- "From C to Java" contains a chapter about refactoring — good to read, but not required for exam.
- Tentative review sheet for final on Web. (But see next slide.)
- Office hours this week to be announced via e-mail soon.

Slide 2

More Administrivia

- Original plan is for the final to be similar to the midterm, but twice as long.
- Alternative: Individual project presentations (about 10 minutes each, worth 50 points), then exam slightly longer than midterm and worth 150 points. Which should we do?

Slide 3

Networking in Java — RMI

- Motivation — for client/server applications, can be annoying to have to design your own protocol.
- Instead, idea is to define “remote objects” that can be treated (at program level) like any other objects — invoke methods.
- Typical use in client/server program:
  - Server creates some remote objects and “registers” them.
  - Clients look up server’s remote objects and invoke their methods.
  - Both sides can pass around references to other remote objects.
- Dynamic code loading possible too.

Slide 4

Networking in Java — RMI, Quick How-To

- Define a class for remote objects:
  - Define interface that extends `Remote`
  - Define class that implements that interface, extends a Java “remote object” class. Can also include other methods, only available locally.
  - Write code using classes — if using as remote object, reference interface; otherwise can reference class.
- Compile and execute:
  - Compile as usual. (Prior to Java 1.5, an extra step was required to generate “stubs” to be used in communicating with remote objects as remote objects.
  - Make classes network-accessible.
  - Start `rmiregistry`.
  - Run server and clients as usual.

Slide 5

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.
  - 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.

Slide 6

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 August to what you know now.

Minute Essay

- How did the course compare to your expectations/goals? Did you learn what you hoped to learn?
- (Also tell me which option you want on the final exam — presentations plus exam, or exam only.)