Slide 1

Administrivia

- Final exam will be option 2 — game presentations plus written exam.
- Homework 8 writeup on Web tomorrow. Grades on earlier homeworks coming soon (by Monday?)

Slide 2

From Quiz 6

- "Which is faster?" question — many people missed.

  (See quiz solution.)

Slide 3

Networking Basics

- Inter-computer communication based on layered approach and "protocols":
  - Application level — HTTP, FTP, telnet, SMTP, POP, IMAP, NTP, etc., etc.
  - Transport level — TCP (Transmission Control Protocol), UDP (User Datagram Protocol).
  - Network level — IP (Internet Protocol — addressing, routing of packets).
  - Link level — device drivers, etc.
- Messages are routed to
  - A machine ("host"), identified by IPA or name.
  - A process, identified by "port number" (16 bits). 0 — 1023 are "well-known ports", others available for applications

Slide 4

Networking Basics —TCP and UDP

- UDP — independent messages, no guarantees about reliability or message order — analogous to (snailmail) letter.
- TCP — point-to-point channel, guarantees reliability and message order — analogous to phone call. Endpoints called "sockets".

Slide 5

Networking in Java

- Classes for communicating at application level — e.g., `URL` ("show URL" example).
- Classes for communicating at network level:
  - TCP — `Socket`, `ServerSocket`.
  - UDP — `Datagram*`.
- RMI (Remote Method Invocation).

Slide 6

Networking in Java —Sockets

- Client/server model:
  - Server sets up "server socket" specifying port number, then waits to accept connections. Connection generates socket.
  - Client connects to server by giving name/IPA and port number — generates a socket.
  - On each side, get input/output streams for socket.

Slide 7

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 8

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, emphplus run `rmi` to generate "stubs" to be used in communicating with remote objects as remote objects.
  - "Make classes network accessibl.
  - Start `rmiregistry`.
  - Run server and clients as usual.

Minute Essay

- Any requests for next Tuesday's class?