Slide 1

Administrivia

- All homeworks up through Homework 6 should be turned in by now, but I'll accept them (probably at reduced credit) through Friday at 5pm.
- Our final is December 14 at 8:30am. A short review sheet is on the Web. Midterms, or at least sample solutions, to be available by Friday.
- Homework 7 and Homework 8 are on the Web, due at the time of the final — no extensions. You're only required to do one of them, but you can do both for extra points. Notice that Homework 7 is worth more points; this means that anyone doing this assignment gets extra points.

Slide 2

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 3

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 4

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 5

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.
  - Very simple example in I/O example from last time.

Slide 6

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 7

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.

Slide 8

Threads in Java

- `Thread` class provides basic functionality. To start a new thread, make a `Thread` object and call its `start` method. Two choices:
  - Create a `Thread` with an object that implements `Runnable` — `run` method has code to execute.
  - Define a subclass of `Thread` that has a `run` method with code to execute.
- Inter-thread interaction based on “monitors” (see o/s or parallel-programming textbooks):
  - Every object (and every class) has a lock.
  - `synchronized` methods must acquire lock — so only one at a time can run.
  - `wait` gives up the lock and sleeps; `notify` and `notifyAll` wake up one/all sleeping thread(s).

Slide 9

- Lots of new stuff in Java 1.5 / 5.0 (`java.util.concurrent` package).

Slide 10

Examples

- For examples of multithreading for performance, multithreading with `wait` and `notify`, refer to my Web site for CSCI 3366 (parallel programming course).
- Formerly many uses for multithreading in GUIs (e.g., animation), but now most can be accomplished with new features of GUI class (e.g., timers).
- Example of socket communication and threading for concurrency — chat example.

Slide 11

Course Recap — What Did We Do?

- Java and object-oriented programming — polymorphism, inheritance, etc.
- 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, exceptions, 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.

Slide 12

Minute Essay

- None — sign in.