Slide 1

Administrivia

- (None?)

Slide 2

Sidebar(?): Controlling Threads in Java

- Preferred method of controlling one thread from another uses "interrupted" status. (Early version of Java provided other methods, e.g., `stop` — now deprecated.)
- Set status with `interrupt` (instance method).
- Check status with `isInterrupted` (instance method) or `interrupted` (static method), or by catching `InterruptedException` thrown by `wait`, `sleep`, `join`, etc.
  (Right — all of these methods potentially throw `InterruptedException`, which is a "checked exception", and in Java you're supposed to do something about those.)
- (Example another time.)

Slide 3

Example Revisited: Generic Master/Worker

- Java version of generic master/worker example not conceptually very different from C-based versions.
- Multithreaded versions use Java `Callable` and `Future`.
- One multithreaded version distributes tasks among worker threads round-robin style, with a shared list of tasks.
- Another multithreaded version sets up a shared queue representing a pool of tasks yet to be done, and worker threads pull items from this pool and process.
- Neither needs an explicit master task.
- (Look at code very briefly?)

Slide 4

Distributed-Memory Programming in Java Using Sockets

- Based on client/server model.
- Before going further, some background . . .

Slide 5

Stream I/O in Java

- Java (of course) allows reading and writing text files, using "streams" that can be connected to standard input/output, files, etc.
- Can also read/write binary data:
  - `DataInputStream`, `DataOutputStream` to write out primitive types.
  - `ObjectInputStream`, `ObjectOutputStream` to write out other types — but only `Serializable` objects.

Slide 6

Object Serialization in Java

- In order to write a non-primitive object to a file, must somehow to turn it into a sequence of bytes; to read, must reconstruct. This is "serialization".
- How does it work?
  - Object and all referenced objects (except `static` and `transient` variables) are turned into sequential stream of bytes.
  - Can override `readObject`, `writeObject` to control what happens more precisely.

Slide 7

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" (and may be off-limits to regular applications), others available for applications.

Slide 8

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 9

Networking in Java

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

Slide 10

Distributed-Memory Programming in Java Using Sockets

- Based on 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, which you can then operate on exactly like you operate on streams connected to files. Program must define protocol for the two sides to communicate. (Like MPI, no? Except you can more easily transmit objects!)

Slide 11

Example: Generic Master/Worker Revisited

- Version using sockets is relatively straightforward — server creates a new thread for each client, only tricky bits are in making sure things are shut down properly. Note use of `synchronized` in code to ensure thread-safe access to shared variables.
- (Look at code. Comments in `Master.java` describe protocol for client/server interaction. Two variants of master process, one expecting fixed number of workers, the other not.)

Slide 12

Minute Essay

- None really — just sign in, unless questions?