Slide 1

Administrivia

- Sample programs page updated. Instructions for installing and using OpenMP compiler added. Homework 2 on Web soon.

Slide 2

Parallel Programming in Java

- Java supports multithreaded (shared-memory parallel) programming as part of the language — `synchronized` keyword, `wait` and `notify` methods of `Object` class, `Thread` class. Programs that use the GUI classes (AWT or Swing) multithreaded under the hood. Justification probably has more to do with hiding latency than HPC, but still useful, and latest version (1.5) includes much new library stuff.

  Aside: If you're curious about thread use in a program that uses AWT or Swing classes, the following line will print info about threads:

  ```
  Thread.currentThread().getThreadGroup().list();
  ```

- Java also provides support for forms of distributed-memory programming, through library classes for networking, I/O (`java.nio`), and Remote Method Invocation (RMI).

Slide 3

What Does A Multithreaded Java Program Look Like?

- Easy answer: Like a regular Java program. (In fact, any program with a GUI . . .)
- Programming model is somewhat like that of OpenMP — all threads share a common address space — but programmer is responsible for creating threads, providing synchronization, etc.

Slide 4

Creating Threads in Java

- Threads are all instances of `Thread` class (or a subclass). Pre-1.5, two ways to create threads:
  - Create a subclass of `Thread` (frowned on by o-o purists).
  - Create a `Thread` using an object that implements `Runnable` (preferable).

  Either way, `run` method (of subclass of `Thread`, or of `Runnable`) contains code for thread to execute.
- Start thread with `start` method. Can wait for it to finish with `join`.
- "Hello world" example (`Hello1.java` and `Hello2.java` on sample programs page).

Slide 5

Shared Variables in Java

- Code executed by a thread is some object's `run` method. Access to variables is consistent with usual Java scoping — class/instance variables, parameters, etc.
- As we noted before, though, simultaneous access to shared variables can be risky, however. So . . .

Slide 6

Synchronization in Java

- Interaction among threads in Java based on "monitor" idea (Hoare (1975) and Brinch Hansen (1975)).
- Every object has implicit lock; `synchronized` keyword means "only run this when you have the relevant lock" — if another thread has the lock, wait. Can be used to ensure one-at-a-time access to critical variables.

  "Relevant lock"? For synchronized methods, lock for object (instance methods) or class (static methods). For synchronized blocks, you specify the object.

  Example — `HelloSynch.java` on sample programs page.
- `wait` and `notify` methods allow more interesting kinds of coordination (next time).

Slide 7

Numerical Integration Example, Revisited

- We all remember the problem. Let's write in Java (`NumInt.java` on sample programs page).

Slide 8

Minute Essay

- What's your username on Sol? (This is so I can check/increase your filesystem quota on Sol, so you can install the OpenMP compiler.)

  (Okay to share pieces of paper on this one.)