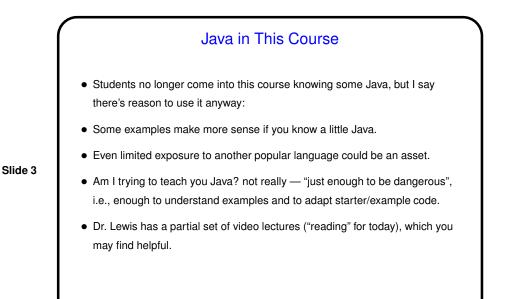
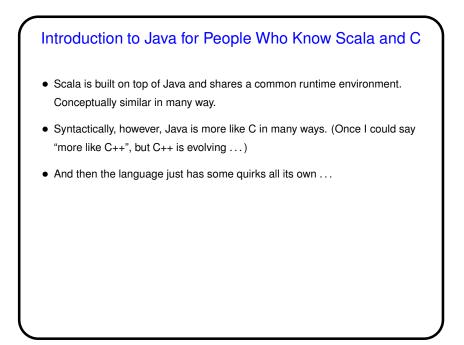


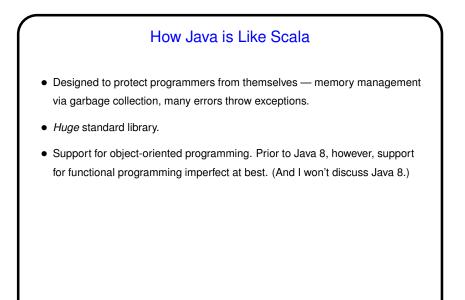
Multithreaded Programming in Java - Overview

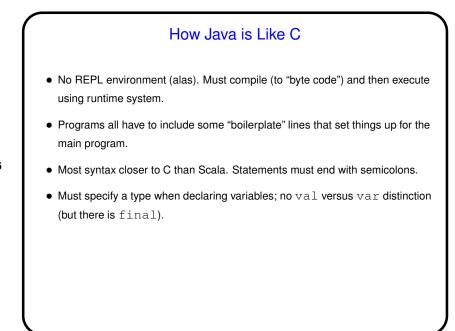
- We'll look next at basic multithreaded programming in Java, mostly focusing on lower-level approaches.
- Why Java? When we wrote the book it was a language many people already knew, including Trinity CS students (after CS2). Now still popular but not used in our required courses. So why ...

Slide 2

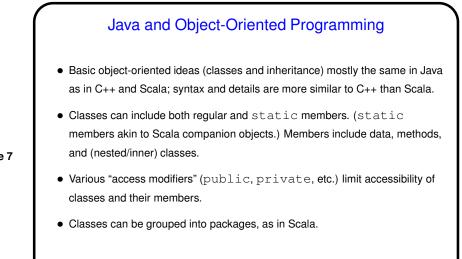








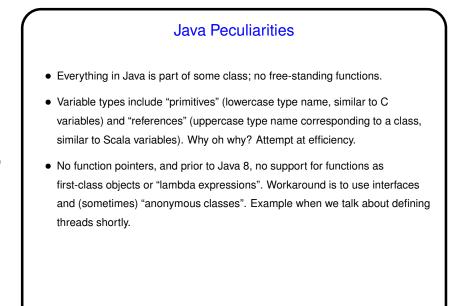
Slide 6



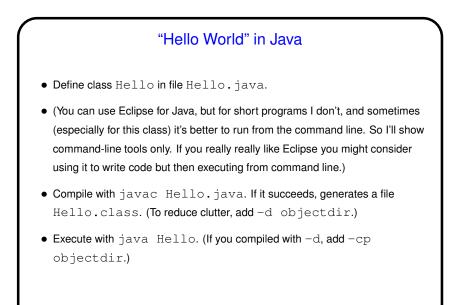
Java and Object-Oriented Programming, Continued

- Type-generic programming possible; syntax more like C++ than Scala.
- No multiple inheritance. C++ has this. Scala doesn't, but allows "traits". Java has "interfaces", akin to limited version of Scala traits. (Traits can have variables, method code; interfaces are purely an API method declarations plus maybe constants.)

Slide 8

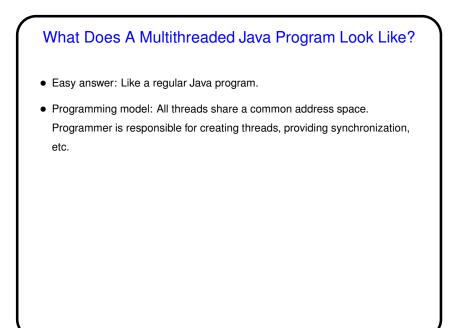


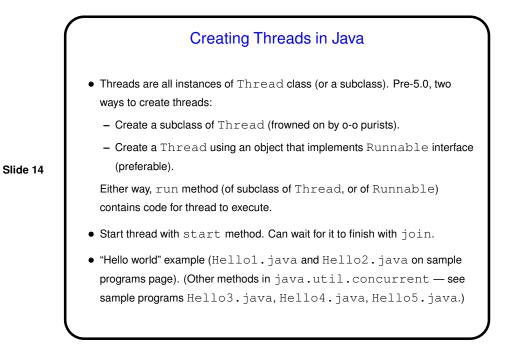
Java Peculiarities, Continued Exception mechanism similar to the one for Scala, *but*: "Unchecked" exceptions can be caught, or not, as you choose. For "checked" exceptions, however, must either catch them or explicitly declare that your method can throw them. Meant to be a good thing — forcing you to think about exceptions that are common enough that you shouldn't just pretend they can't happen — though in practice sometimes annoying. Compiler picky about names: Only one public class per file, and name of file must match name of class. Runtime system picky about directory structure; must match package structure.

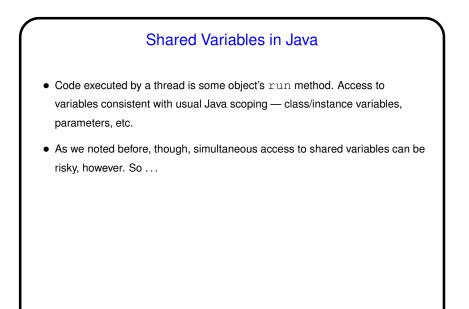


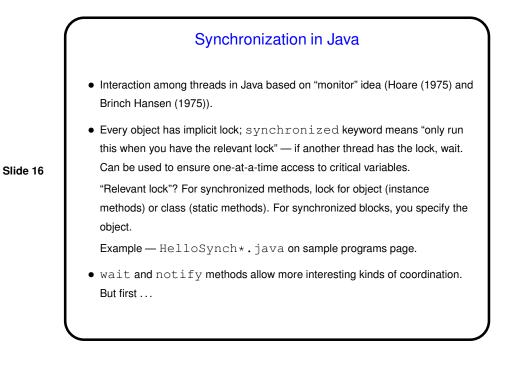
- Parallel Programming in Java
- Java supports multithreaded (shared-memory parallel) programming as part of the language — synchronized keyword, wait and notify methods of Object class, Thread class. Programs that use GUI libraries multithreaded under the hood. (Scala shares this property.) Justification probably has more to do with hiding latency than HPC, but still useful, and versions 5.0 and beyond includes much useful library stuff.
- Java also provides support for forms of distributed-memory programming, through library classes for networking, I/O (java.nio), and Remote Method Invocation (RMI).

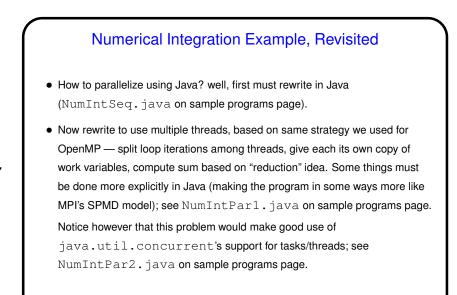
Slide 12











Synchronization in Java, Continued
synchronized methods/blocks can be used to ensure that only one thread at a time accesses some shared variable.
For more complex synchronization problems, can use wait and notify (or notifyAll):

wait suspends executing thread (adds to "wait set").
notify wakes up one thread from the wait set. notifyAll wakes up all threads in the wait set. Newly-awakened thread(s) then compete to reacquire lock and continue execution.
Can only be done from within synchronized method/block.
Typical idiom — loop to check condition, wait.
(More about this, and example, later.)

Slide 17