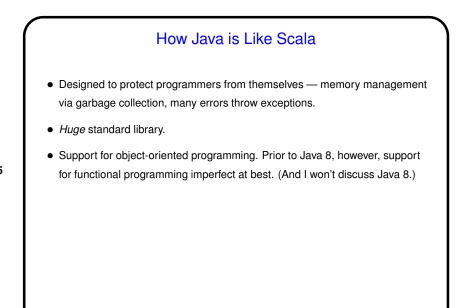
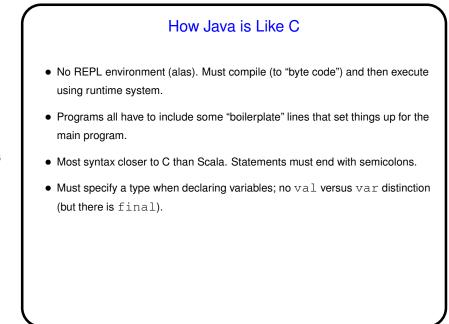
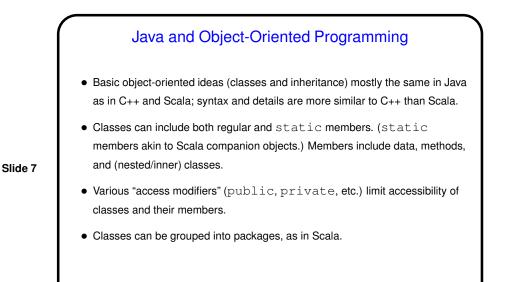


Introduction to Java for People Who Know Scala and C
Scala is built on top of Java and shares a common runtime environment. Conceptually similar in many way.
Syntactically, however, Java is more like C in many ways. (Once I could say "more like C++", but C++ is evolving ...)
And then the language just has some quirks all its own ...

Slide 3



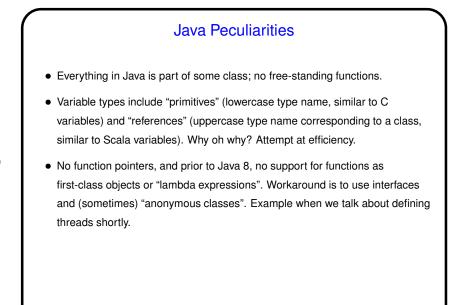




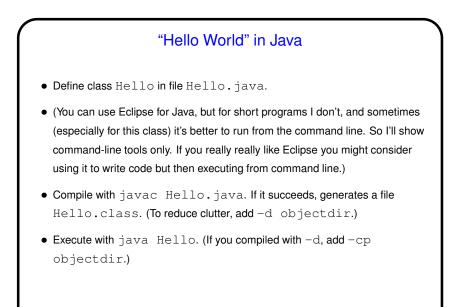
Java and Object-Oriented Programming, Continued

- 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

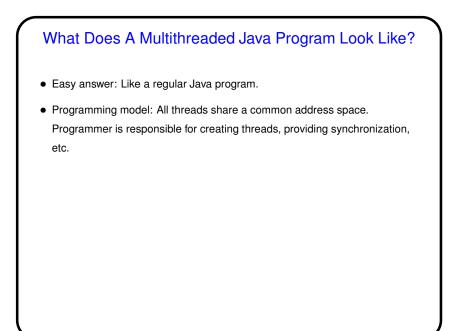


Java Peculiarities, Continued Exception mechanism similar to the one for Scala, *but*: "Unchecked" exceptions can be caught, or not, as you choose. For "checked" exceptions, however, must either catch them or explicitly declare that your method can throw them. Meant to be a good thing — forcing you to 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).



Creating Threads in Java Threads are all instances of Thread class (or a subclass). Pre-5.0, 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 interface (preferable). Slide 14 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 (Hellol.java and Hello2.java on sample programs page). (Other methods in java.util.concurrent — see sample programs Hello3.java, Hello4.java, Hello5.java.)

