Java: make it run, make it right, make it fast

(see Byte, May 1998, for more details)

- "Java isn’t fast enough for ‘real’ applications" --- J. Generic
  - Moore’s law: hardware speeds double every 18 months
  - What’s the analogous law for software?

- Java: .java to .class to JVM to executing code, where is there room to make this faster?
  - Source to bytecode optimizers, currently primitive given state-of-the-art compiler technology e.g., in C++ compilers
  - bytecode: post compilation re-ordering and optimizing
  - better JVMs, better garbage collection, thread management
    - Sun/Microsoft generational garbage collection: “most objects die young"
Java: making it fast (continued)

- **JIT: just-in-time compiling**
  - as Java class files are loaded, they are compiled to native (platform-specific) code and cached for continued fast execution (Netscape and Symantec’s JIT)
  - non-JIT JVM: 50% of time executing bytecode (what else?)

- **Dynamic/Adaptive compilation**
  - Sun: HotSpot, compile bytecode to native code at run time based on execution profiles

- **Java Chips**
  - execute bytecode at the machine level
Java: make it fast by smart/better programming

- **Strings are really slow**
  - use `StringBuffer` or char array (when you know there’s a bottleneck). String concatenation is egregiously slow: convert to buffer, join, convert back
- **Use a good ClassLoader, e.g., JBuilder (Borland) is faster than Sun’s loader**
- **Avoid temporary objects, lots of work for garbage collector**
- **Use static initialization when appropriate, e.g., as a replacement for some parts of a constructor**
- **Use JFC instead of AWT, better thread management, no peers**

- **Profile your code, don’t guess**
CPS 108

- On becoming an object-oriented programmer and designer
  - C++, Java, similarities and differences
  - Using inheritance wisely, knowing about design patterns
  - conventions, e.g., UML, design methodologies
- More knowledge of C++ and it’s C core
  - some low level details, knowing what to expect in a C program
- Knowledge of Java
  - comparison to C++, which language will you turn to?
- Working in teams/groups on bigger programs
  - managing teams, working together, using tools like CVS
- The software process
  - how to make good programs, how to deliver good products