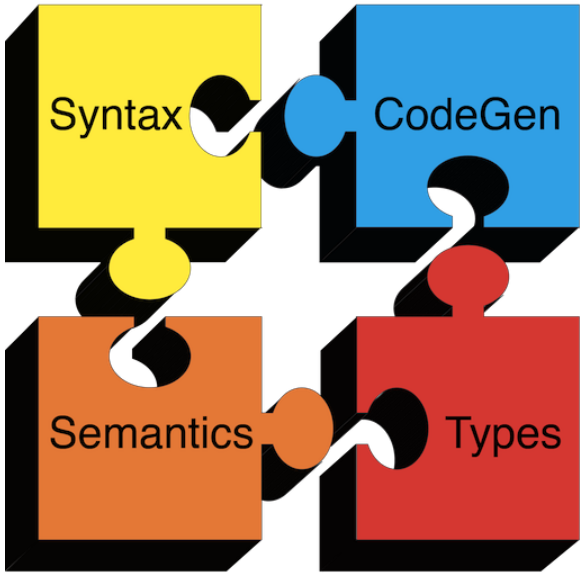

Compilers

Other Topics

Other Topics

- Java allows classes to be loaded at run time
  - Type checking source takes place at compile time
  - Bytecode *verification* takes place at run time
- Loading policies handle by a ClassLoader
- Classes may also be unloaded
  - Not well-specified in the definition

Other Topics

- Initialization in Java is complex
  - Everything in COOL plus much more
  - Greatly complicated by concurrency

- A class is initialized when a symbol in the class is first used
  - Not when the class is loaded
  - Delays initialization errors to a predictable point (when something in the class is referenced)

Other Topics

1. Lock the class object for the class
   - Wait on the lock if another thread has locked it
2. If the same thread is already initializing this class, release lock and return
3. If class already initialized, return normally
4. Otherwise, mark initialization as in progress by this thread and unlock class

5. Initialize superclass, fields (in textual order)
   - But initialize static, final fields first
   - Give every field a default value before initialization
6. Any errors result in an incorrectly initialized class, mark class as erroneous
7. If no errors, lock class, label class as initialized, notify threads waiting on class object, unlock class

Other Topics

- In any system with N features, there are potentially N^2 feature interactions.

- Big, featureful systems are hard to understand!
  - Including programming languages

Other Topics

- Java is well done
  - By production language standards, very well done
- Java brought many important ideas into the mainstream
  - Strong static typing
  - Garbage collection
- But Java also
  - Includes features we don’t fully understand
  - Has a lot of features