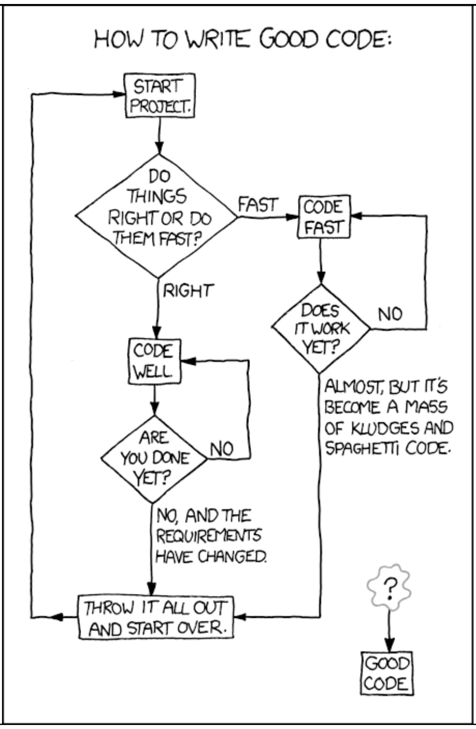
Objectives

- Software Quality Metrics
- Static Analysis Tools
- Refactoring for Extensibility

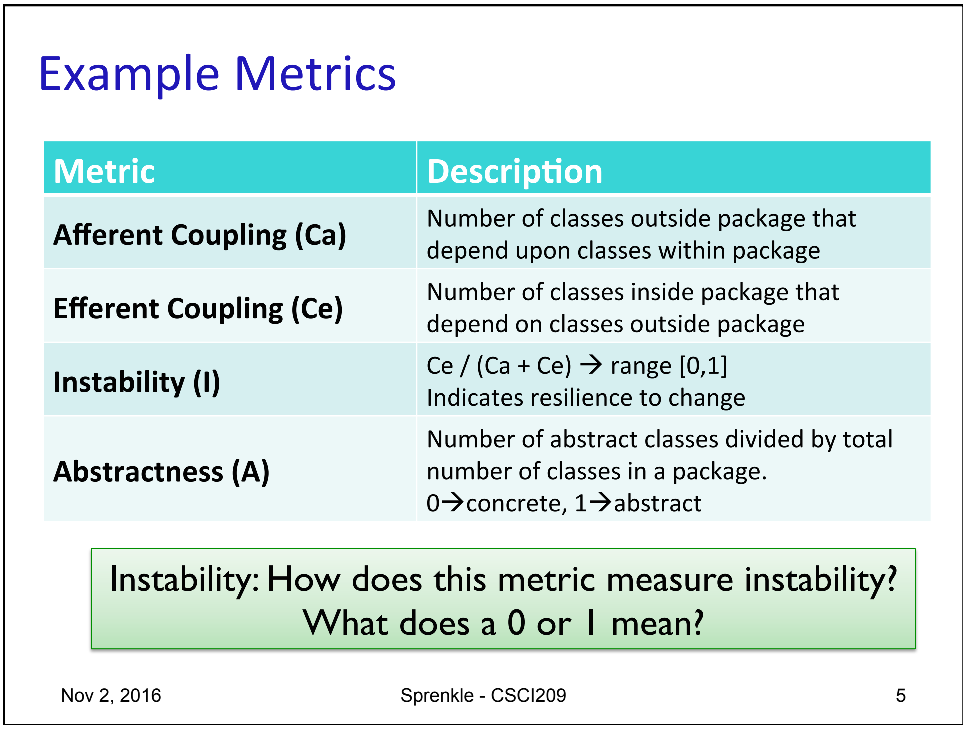

HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL
ARE YOU DONE YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

So what does this mean for us?

SOFTWARE QUALITY METRICS

Metrics to Measure Software Quality

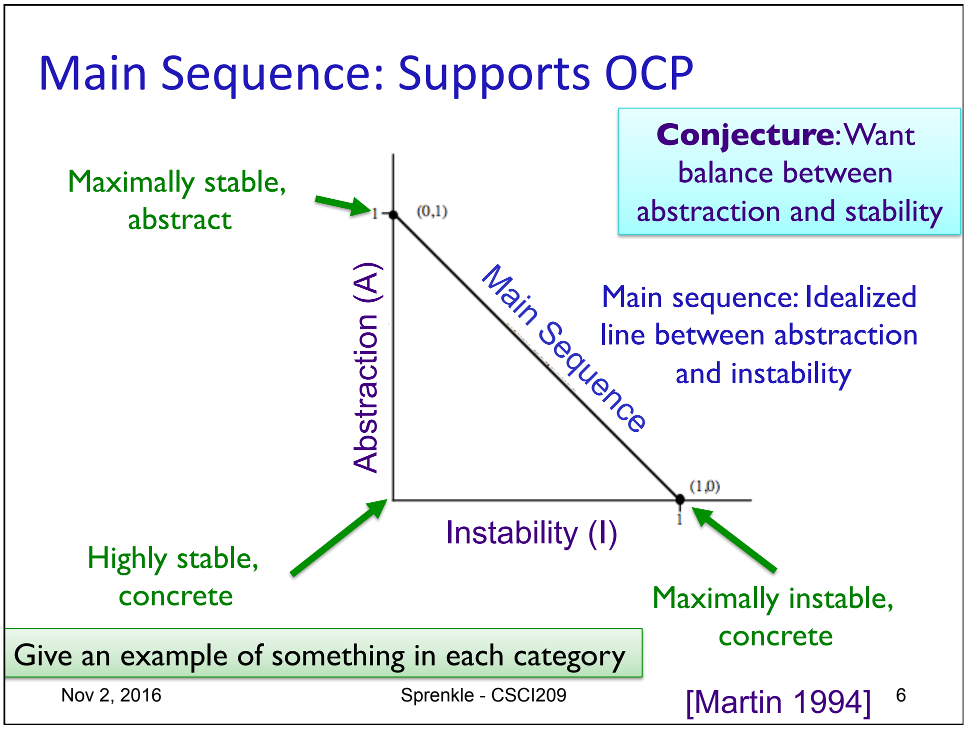
- Create metrics to help us figure out if our code is good and what we can improve
  - Add a little more science
- Examples: number of methods, # loc /method, # attributes/class
- Tricky: Not clear what is "good" number
  - Requires good judgment, experience
  - Metrics often should not be considered in isolation



Example Metrics

Metric	Description
Afferent Coupling (Ca)	Number of classes outside package that depend upon classes within package
Efferent Coupling (Ce)	Number of classes inside package that depend on classes outside package
Instability (I)	Ce / (Ca + Ce) → range [0,1] Indicates resilience to change
Abstractness (A)	Number of abstract classes divided by total number of classes in a package. 0→concrete, 1→abstract

Instability: How does this metric measure instability? What does a 0 or 1 mean?



Main Sequence: Supports OCP

Conjecture: Want balance between abstraction and stability

Maximally stable, abstract

Main sequence: Idealized line between abstraction and instability

Highly stable, concrete

Maximally instable, concrete

Give an example of something in each category

[Martin 1994]

Example: Lack of Cohesion of Methods (LCOM)

- A measure of a class's *cohesiveness*
- Calculated with the Henderson-Sellers method:
  - m(A) : # of methods accessing an attribute A
  - Calculate the average of m(A) for all attributes, subtract the number of methods m, and divide the result by (1-m)

$$\frac{\overline{m(A)} - \text{\# of methods}}{1 - \text{\# of methods}}$$

Analysis and Discussion:
What does LCOM tell us?

$$\text{LCOM} = \frac{\overline{m(A)} - \text{\# of methods}}{1 - \text{\# of methods}}$$

m(A) is # methods accessing attribute A

- What is the relationship between m(A) and # of methods?
- What are the extremes?
  - Every method accesses every attribute?
  - Every attribute is accessed by one method?

Plugins

- Tons of available plug-ins
- Some better than others
  - Haven’t been updated for awhile
  - Aren’t highly ranked
- Documentation is often lacking
  - Look for new plugin in various menus, including Properties option, Views, Perspectives
- Try out new plugins
  - if you don’t like it, you can uninstall

Eclipse Metrics Plugin

- Provides information about your classes
  - # of classes
  - # of lines of code per method
  - # of attributes
  - Coupling (afferent, efferent)
  - Instability
  - …
- Update site:
  - `http://metrics2.sourceforge.net`

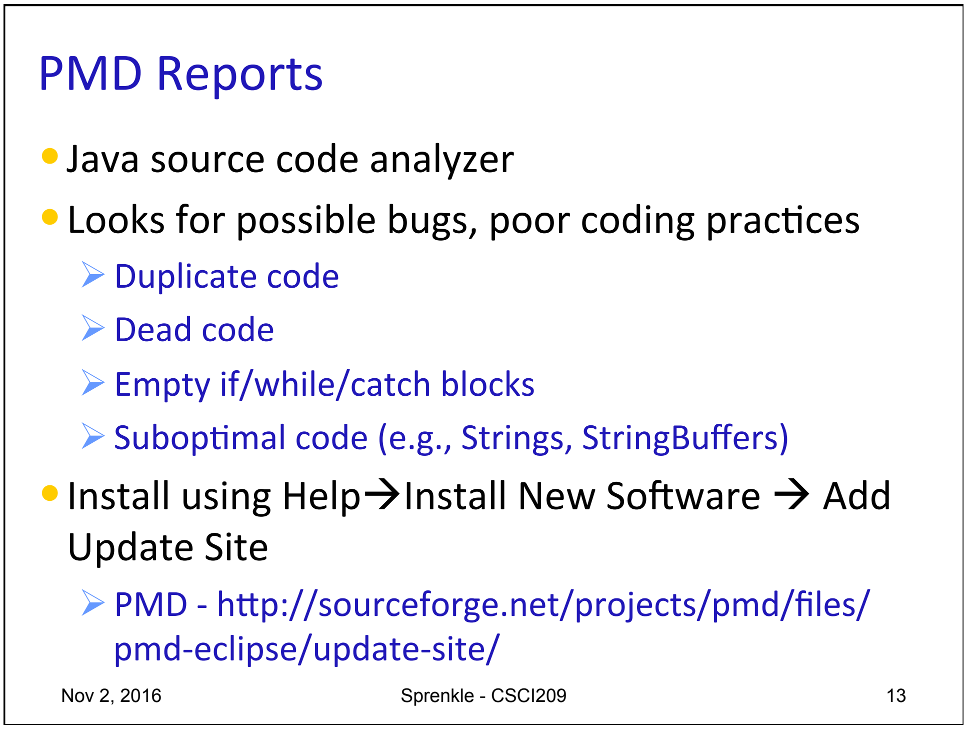
FindBugs

http://findbugs.sourceforge.net/

- Performs static analysis to look for bugs
  - Looks for "bug patterns"
- Types of errors
  - Correctness
  - Bad practice
  - "Dodgy" → style
- Eclipse Marketplace
  - Under `Help` menu
  - Search for FindBugs

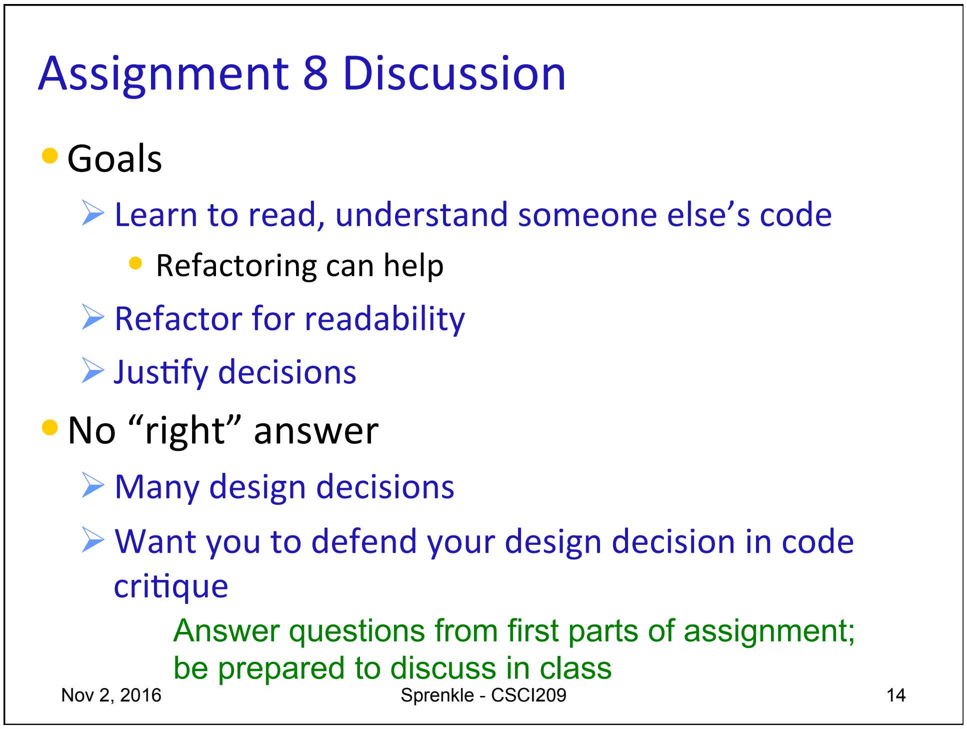
Checkstyle Plugin

- Checks that Java code adheres to a set of coding standards
- Examples:
  - Identifies variables that should be declared final
  - Identifies code that is not open to extension and how to handle
  - Will report issues with tabbing/spacing
- Install through Eclipse Marketplace



PMD Reports

- Java source code analyzer
- Looks for possible bugs, poor coding practices
  - Duplicate code
  - Dead code
  - Empty if/while/catch blocks
  - Suboptimal code (e.g., Strings, StringBuffers)
- Install using Help→Install New Software → Add Update Site
  - PMD - http://sourceforge.net/projects/pmd/files/pmd-eclipse/update-site/



Assignment 8 Discussion

- Goals
  - Learn to read, understand someone else’s code
    - Refactoring can help
  - Refactor for readability
  - Justify decisions
- No “right” answer
  - Many design decisions
  - Want you to defend your design decision in code critique

Answer questions from first parts of assignment; be prepared to discuss in class