Objectives

- Metrics
  - Metrics plugin
- Refactoring for Extensibility

Review

- What property are we designing to?
- What is the typical fix for designing more flexible/maintainable code?
  - What is a limitation of that?

Notes on Assignment 10

- No "right" answer
  - Many design decisions
  - Want you to defend your design decision in code critique

Summary of Designing for Change

Use ***abstraction*** for code that is *likely to change*

- Can depend on code that is *stable* and unlikely to change
  - Example of stable code: `System.out`

METRICS

Metrics to Measure Software Quality

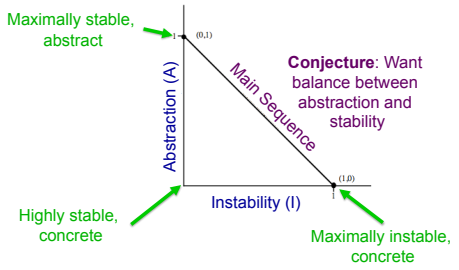
- Create metrics to help us figure out if our code is good and what we can improve
  - Add a little more "science"
- Examples: number of methods, # loc / method, # attributes/class
- Tricky: Not clear what is "good" number
  - Requires good judgment, experience
  - Metrics often should not be considered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



[Martin 1994]

Example: Lack of Cohesion of Methods (LCOM)

- A measure of a class's *cohesiveness*
- Calculated with the Henderson-Sellers method:
  - If m(A) is the number 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?

Example: Lack of Cohesion of Methods (LCOM)

- A measure of a class's *cohesiveness*
- Calculated with the Henderson-Sellers method:
  - If m(A) is the number 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)
- Low value → a cohesive class
- Value close to 1 → a lack of cohesion
  - Suggests class might better be split into a number of (child) classes

Metrics Plugin

- Install plugin: Help menu -> Software Updates -> Find and Install
  - New Remote Site
    - Name: `Metrics`
    - URL: `http://metrics.sourceforge.net/update`

Metrics Plugin

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

REFACTORING FOR EXTENSIBILITY

Simulating a Roulette Game

- See handout

Get a Solution

- Import Existing Project
  - `/home/courses/cs209/handouts/roulette.tar`

Understanding Code

- Execute the code
  - What is the main driver for this project?
- What are each class's responsibilities?
- What does `test.RouletteTestSuite` do?

Bug in the Code

- Determining if Odd/Even Bet was won is incorrect

Understanding Code

- Focus: how **open** is the code to adding **new** kinds of **bets** and how **closed** it is to **modification**?
  - How many classes know about the `Bet` class?
  - What code would need to be added to `Game` to allow the user to make another kind of bet that paid one to one odds and was based on whether the number spun was high (between 19 and 36) or low (between 1 and 18)?

Assignment 10 Due on Friday