Software Construction

- More than just programming
- Involves design
- Team organization
  - Who works on what?
    - Often matches code organization/modularization
  - SCM implications
Construction Best Practices

• Assertive programming
  – Test-driven development
  – Design-by-Contract
• Refactoring
• Communication
  – Reviews, pair-programming, stand-up meetings
• Know one development environment well
• Coding standards
Refactoring

• Software is more like gardening than construction
• Refactoring: changing the internal structure of code without changing its external behavior
  – Don’t try to refactor and add functionality at the same time
  – Have good tests and run them often when refactoring
  – Take short, deliberate steps
• Become familiar with automated refactoring tools
• See www.refactoring.com
Pragmatic Programmer: Tip 4

• Don’t Live with Broken Windows
• What are Broken Windows in Software?
  – Bad designs, wrong decisions, poor code, no tests; beginning of software entropy
• If you don’t have time to fix it what do you do?
  – Board it up, i.e. remove it or declare it as “not implemented”
Pragmatic Programmer: Tip 11

- DRY - Don’t Repeat Yourself
- Avoid cut-and-paste
  - Refactor to reusable code instead
Minimize Complexity and Coupling; Maximize Cohesion

• Complexity
  – Organize into modules
    • Abstract at all levels
  – Keep code simple and readable

• Coupling
  – Use interfaces, Law of Demeter

• Cohesion
  – Keep methods short
  – Don’t let classes get too big
Law of Demeter

• Any method of an object should call only methods belonging to:
  – Itself
  – Any parameters that were passed in to the method
  – Any objects it created
  – Any directly held component objects
• i.e. don’t use objects to get other objects