Lectures on Martin Fowler
Refactoring Methodology

Part 1: Bad Smells in Code

Rushikesh K. Joshi
Duplicate Code

- Many similar looking code sections
- As program evolves with new functionality
  - Copy edits happen
  - Lazy programmers
  - Inability to conceptualize and parameterize
- Solution strategies:
  - Extract Method
  - Parameterize
  - Template Methods
Long Method

- Longer the method, more difficult it is to understand
- Many parameters and temporary variables pose difficulties in solving this problem
- Solutions strategies:
  - Extract method
  - Parameterize
Large Class

• Class is trying to do too much
• Probably less cohesive
• Solution strategies:
  – Extract class
  – Extract subclass
Long parameter List

• Probably not object oriented
• Too general
• Solution strategies:
  – Objectify the values by introducing parameter object
  – Send the object itself in, from which the values are taken in
Divergent Change

• Can't easily change the class when it needs to be modified
  – Many changes needed in the class to get something new
  – Possible due to copy-edit code etc.
    • e.g. everytime you change data, you may have to change a couple of methods.. all the time

• Solution strategies:
  – Express the variation
    • Extract superclass/subclass
  – Extract class, Parameterize
Shotgun Surgery

• When a change is made (to a class), it affects many other classes

• Opposite of divergent change

• Solution strategies:
  – Move method(s), move field(s) to concentrate all changes into one place
  – Extract class, Parameterize
  – Achieve one-to-one correspondance between changes and classes that get affected
Feature Envy

- A method is more interested in a class external to its own class
- High coupling as compared to cohesion
- If it's coupled with many classes, which one is the best suited class for it?

Solution strategies:
- Analyze coupling
- Extract and Move method
- Visitor or related patterns may be useful
Data Clumps

• Several data fields flocking together
  – In parameters
  – As locals

• Solution strategies:
  – Form an object out of them, using extract class, preserve whole object
Primitive Obsession

• Too many primitive data types such as those in Java
  – New programmers use them instead of using objects

• Solution strategies:
  – Replace data values with object structures
  – Make types with the help of classes e.g. use array objects
Switch Statements

• Several switch statements occur together
  – If one has to be changed, many have to be
  – Show dependence on object types

• Solution strategies:
  – Use polymorphism
  – Extract method, make subclasses
Parallel Inheritance Hierarchies

- Everytime a subclass is added into one hierarchy, you have to add one more into another

- Solution strategies:
  - Use instances of one hierarchy into another
  - Move method, move field
Lazy Class

• Class that is not being used enough, one that is too small and can be inlined

• Solution strategies:
  – Inline class
  – Collapse hierarchy
Sepculative Generality

- Too much future planning that is not being used
- Adds to complexity, get rid of it

Solution strategies:
- Remove parameter
- Collapse hierarchy
Temporary Fields

• Local temporary variables of a method are made as instance variables in the class

• Solution strategies:
  – Move fields into methods
  – Make temporary objects
Message Chains

- Complex interaction between objects by means of delegation from into another
- Solution strategies:
  - Extract method to do the orchestration
Middle Man

• An intermediate is used for some purpose

• Solution strategies:
  – Add into hierarchy
  – Inline into caller
Inappropriate Intimacy

- Accessibility into private fields
- Coupling breaking encapsulation
- Overuse of friend relation

Solution strategies:
- Move field
- Move method
- Inner classes
Alternative Classes with Different Interfaces

- Methods doing same thing with different signatures
- They may be similar but still not enough to pull into one hierarchy
- Solution strategies:
  - Rename methods to get overloading
  - Complete the methods to pull into hierarchy, extract superclass
Incomplete Library Class

• Functionality on library classes has to be developed externally

• Solution strategy:
  – Introduce foreign method
    • A method in client class with an instance of the server
  – Introduce local extension
    • A local extension of the server class
Data Class

- Just get/set classes without much functionality
- Fields are public

Solution strategies:
- Encapsulate
- Move method to move data-use methods from elsewhere into this class
Refused Bequest

• Subclasses don't need methods from parents
• Solution strategies:
  – Push down method
  – Push down field
  – Replace inheritance with delegation
Comments

- Too many comments may indicate badly written code!

- Solution strategies:
  - Extract method
  - Rename method
  - Rename parameters
  - Introduce assertions