CS 208
Software Engineering

Refactoring

Abhishek Srivastava
Office: PS-02 (C), PACL Campus
Indian Institute of Technology
Phone: (07324)240769
Email: asrivastava@iiti.ac.in

Refactoring

[Fowler 1999]:

- change a software system so that the external behaviour does not change but the internal structure is improved

- improve the design of existing code in small steps

- "If it stinks, change it."
  [Kent Beck]

When to refactor?

- when you plan to add functionality
  - make the code more suited for the new addition
- when you need to find bugs
- make the code more clear to understand

When not to refactor?

- when you should rewrite
- when you are close to a deadline

Bad smells in code

Duplicated code

- The same functionality in more than one place
  - e.g., same expression in two methods of the same class
  - e.g., same expression in two sibling subclasses

- Apply Extract Method, Pull Up Method

Bad smells ...

Long Method

- Long, difficult-to-understand methods

- Want short, well-named methods
- Aggressively decompose methods
  - reusable blocks of code
- Write a method instead of a comment

- Apply Extract Method

Bad smells ...

Large Class

- A class trying to do too much
  - e.g., too many instance variables

- Needs to be split
- Separation of concerns

- Apply Extract Class to factor out some related set of variables and methods

Bad smells ...

Long Parameter List

- Passing in lots of parameters to a method (because "globals are bad")
- Difficult to understand

- Pass only enough so that the method can get to everything it needs
- Pass objects

Bad smells ...

Divergent Change

- When one class is commonly changed in different ways for different reasons
- No clear point for where to make changes leads to degradation

- Perhaps two (or more) classes are better than one
- Apply Extract Class

Bad smells ...

Shotgun Surgery

- Making a change requires many little changes to many different classes
- The opposite of divergent change

- Consolidate the changes to one class (Inline Class)
- Balance with divergent change

- Apply Move Method and Move Field

Bad smells ...

Feature Envy

- A method seems more interested in the details of a class other than the one it is actually defined in
  - e.g., invoking lots of get methods

- Apply Move Method

Bad smells ...

Data Clumps

- Groups of data appearing together in the fields of classes, parameters to methods, etc.
  - e.g., int x, int y, int z
- Move these groups into their own class
- Apply Extract Class and Introduce Parameter Object

Bad smells ...

Primitive Obsession

- Using the built-in types of the language too much
  - e.g., telephone numbers are represented as a string
- Reluctance to use small objects for small tasks

- Use objects for individual data values

- Apply Replace Data Value with Object

Bad smells ...

Lazy Class

- A class that is not doing enough to “pay”its own way
- Could be eliminated
- Apply Collapse Hierarchy or Inline Class

Bad smells ...

Speculative Generality

- “I think we might need this someday.”
  - e.g., abstract classes without a real purpose
  - e.g., unused parameters
- Increases design complexity unnecessarily

- Apply Collapse Hierarchy and Remove Parameter

Bad smells ...

Middle Man

- A class that delegates many methods to another class
- Apply Remove Middle Man

Bad smells ...

Inappropriate Intimacy

- Two classes that depend too much on each other, with lots of bidirectional communication
- Separate the two classes
- Apply Move Method, Move Field, and Extract Class (factor out commonality)

Bad smells ...

Alternative Classes with Different Interfaces

- Methods that do the same thing but have different signatures
  - e.g., put() versus add()
- Want software interfaces to be "consistent"
- Apply Rename Method

Bad smells ...

Data Class

- Classes that are all data (manipulated by other classes with getters/setters)
    - e.g., a Point record that has other classes manipulating its coordinates
- Study usage and move appropriate behavior into data classes

- Apply Encapsulate Field, Extract Method, Move Method

Bad smells ...

Refused Bequest

- When a subclass inherits something that is not needed
- When a superclass does not contain truly common state/behaviour

- Can use Push Down Method and Push Down Field

Bad smells ...

Comments

- Often just "deodorant" for bad smelling code
- Refactor code so that the comment becomes extraneous

- Comments are good for explaining why you did something