



Bad Smells in Code

Eduardo Figueiredo

<http://www.dcc.ufmg.br/~figueiredo>

[Bad Smell]

- Bad smell is any symptom in the source code of a program that possibly indicates a problem
- Determining what is and what is not a bad smell is subjective
 - It varies by languages, developers and development methodologies



List of Bad Smells

Refused Request	Large Class	Long Method	Comments
Divergent Change	Shotgun Surgery	Feature Envy	Long Parameter List
Primitive Obsession	Switch Statements	Lazy Class	Speculative Generality
Temporary Field	Message Chains	Middle Man	Inappropriate Intimacy
Duplicated Code	Data Clumps	Data Class	Incomplete Library Class
Parallel Inheritance Hierarchies		Alternative Classes with Different Interfaces	

[Duplicated Code]

- The same code structure in more than one place
 - Your program is always better when you avoid duplicated code
- Candidate refactorings
 - *Extract Method*: create a new method with the duplicated code
 - *Pull Up Method*: move the general method to a superclass



[Long Method / God Method]

- A method that centralizes the behavior of a class
 - The longer a method is, the more difficult it is to understand
- Candidate refactorings
 - *Extract Method*: split a method into two
 - *Replace Method with Method Object*: turn the method into its own class

[Large Class / God Class]

- A class doing too much
 - Symptoms are too many attributes and too much code
- Candidate refactorings
 - *Extract Class*: split a class into two classes
 - *Extract Subclass*: create a subclass of the give class

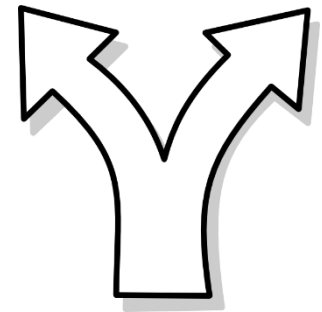


[Long Parameter List]

- Everything is passed as parameter in structured programming to avoid global variables
 - OOP changed this practice
- Candidate refactoring
 - *Introduce Parameter Object*: replace several parameters by an object

[Divergent Change]

- It occurs when one class is changed in different ways for different reasons
 - A class should change as a result of only one kind of modification
- Candidate refactorings
 - *Extract Class*: split a class into two classes



[Shotgun Surgery]

- It is the opposite of Divergent Change
 - Every time you make a change, you have to make a lot of little changes to a lot of different classes
- Candidate refactorings
 - *Move Method* and *Move Field*: put closer methods and attributes that change together
 - *Inline Class*: merge two classes in one class

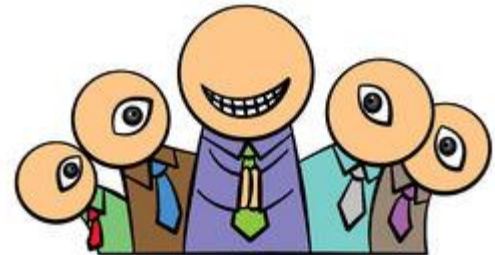


[Feature Envy]

- A code fragment more interested in a class other than the one it is in
 - Example: a method that only accesses attributes of another class

- Candidate refactorings

- *Move Method* and *Move Field*: put methods and attributes closer
- *Extract Method*: if only part of a method suffers from Feature Envy



[Switch Statements]

- The problem with switch statements is similar to duplicated code
 - You often find the same structure of switch in different parts of your program
- Candidate refactorings
 - *Extract Method*: create a new method for each case
 - *Replace Conditional with Polymorphism*: to set up the inheritance structure

[Lazy Class]

- Each class costs money to maintain and to understand
 - Lazy class is not doing enough to pay for itself
- Candidate refactorings
 - *Collapse Hierarchy*: merge a subclass with its superclass
 - *Inline Class*: merge two classes



[Refused Bequest]

- A subclass inheriting methods and data, but it does not need them
 - The hierarchy is probably wrong
- Candidate refactorings
 - *Push Down Method / Field*: move from superclass to appropriate subclass(es)
 - *Replace Inheritance with Delegation*: change a inheritance relationship by a delegation

[Comments]

- Comments are not a bad smell
 - Comments are often used as a deodorant
 - Comments are written when code is bad
- Candidate refactorings
 - *Extract Method*: create a new method for a block of commented statements
 - *Rename Method/Field*: give good names for methods and attributes

[Bibliography]

- Martin Fowler. **Refactoring: Improving the Design of Existing Code**. Addison-Wesley Professional, 1st edition, 1999.
 - Chapter 3 - Bad Smell in Code