Smells Within Classes

Smell	Description	Refactorings
Comments	Should only be used to clarify "why" not "what". Can quickly become verbose and reduce code clarity.	Extract Method Rename Method Introduce Assertion
Long Method	The longer the method the harder it is to see what it is doing.	Extract Method Replace Temp with Query Introduce Parameter Object Preserve Whole Object Replace Method with Method Object
Long Parameter List	Don't pass in everything the method needs; pass in enough so that the method can get to everything it needs.	Replace Parameter with Method Preserve Whole Object Introduce Parameter Object
Duplicated Code		Extract Method Pull Up Field Form Template Method Substitute Algorithm
Large Class	A class that is trying to do too much can usually be identified by looking at how many instance variables it has. When a class has too many instance variables, duplicated code cannot be far behind.	Extract Class Extract Subclass
Type Embedded in Name	Avoid redundancy in naming. Prefer schedule.add(course) to schedule.addCourse(course)	Rename Method
Uncommunicative Name	Choose names that communicate intent (pick the best name for the time, change it later if necessary).	Rename Method
Inconsistent Names	Use names consistently.	Rename Method
Dead Code	A variable, parameter, method, code fragment, class, etc is not used anywhere (perhaps other than in tests).	Delete the code.
Speculative Generality	Don't over-generalize your code in an attempt to predict future needs.	If you have abstract classes that aren't doing much use Collapse Hierarchy Remove unnecessary delegation with Inline Class Methods with unused parameters - Remove Parameter Methods named with odd abstract names should be brought down to earth with Rename Method

Smells Between Classes

Smell	Description	Refactorings
Primitive Obsession	Use small objects to represent data such as money (which combines quantity and currency) or a date range object	Replace Data Value with Object Replace Type Code with Class Replace Type Code with Subclasses Replace Type Code with State/Strategy If you have a group of fields that should go together, use Extract Class If the primitives occur in a param lists use Introduce Parameter Object When working with an array consider Replace Array With Object
Data Class	Classes with fields and getters and setters and nothing else (aka, Data Transfer Objects - DTO)	Move in behavior with Move Method
Data Clumps	Clumps of data items that are always found together.	Turn the clumps into an object with Extract Class Then continue the refactoring with Introduce Parameter Object or Preserve Whole Object
Refused Bequest	Subclasses don't want or need everything they inherit. The Liskov Substitution Principle (LSP) says that you should be able to treat any subclass of a class as an example of that class.	Most of the time that's fine, just don't use what you don't need. Occassionally you'll need to create a new sibling class and use Push Down Method and Push Down Field The smell is worst if a subclass is reusing behavior but does not want to support the interface of the superclass. In this case use Replace Inheritance with Delegation
Inappropriate Intimacy	Two classes are overly entertwined.	Move Method Move Field Change Bidirectional Association to Unidirectional Association Extract Class Hide Delegate Replace Inheritance with Delegation
Lazy Class	Classes that aren't doing enough should be refactored away.	Collapse Hierarchy Inline Class
Feature Envy	Often a method that seems more interested in a class other than the one it's actually in. In general, try to put a method in the class that contains most of the data the method needs.	Move Method May need to use Extract Method first, then Move Method
Message Chains	This is the case in which a client has to use one object to get another, and then use that one to get to another, etc. Any change to the intermediate relationships causes the client to have to change.	Hide Delegate Or try using Extract Method and then Move Method to move it down the chain.
Middle Man	When a class is delegating almost everything to another class, it may be time to refactor out the middle man.	Remove Middle Man If only a few methods aren't doing much, use Inline Method You could also consider turning the middle man into a subclass with Replace Delegation with Inheritance
Divergent Change	Occurs when one class is commonly changed in different ways for different reasons. Any change to handle a variation should change a single class.	Identify everything that changes for a particular cause and use Extract Class to put them all together.
Shotgun Surgery	The opposite of Divergent Change. A change results in the need to make a lot of little changes in several classes.	Use Move Method and Move Field to put all the changes into a single class. Often you can use Inline Class to bring a whole bunch of behavior together.
Parallel Inheritance Hierarchies	A special case of Shotgun Surgery. Every time you make a subclass of one class, you also have to make a subclass of another.	Use Move Method and Move Field to combine the hierarchies into one.

This page originally appeared at http://wiki.java.net/twiki/bin/view/People/SmellsToRefactorings