Symptoms and Smells

Symptoms	Smell
Duplication	
Methods and/or classes have similar behavior but different names	Alternative Classes with Different Interfaces
Similar code	Duplicated Code
Code with similar effects	
Names	
Two different names for the same thing	Inconsistent Names
Name is compound word, including a type name	Type embedded in Name
Hungarian notation (type indicator used as prefix)	
Variable named after type rather than intent	
One- or two-character name	Uncommunicative Name
Name without vowels	
Numbered variables	
Odd abbreviations	
Name that doesn't reflect its real use or meaning	
Code within a Method	
// or /* */	Comments
Complex expression using `&&, ||, !`	Complicated Boolean Expression
Large number of lines in method	Long Method
Large number of parameters to method	Long Parameter List
Constant embedded in code	Magic Number
Comparison against `null`	Null Check
Complex if statement	Special Case
Conditional test before body of code	

`switch` keyword used	Simulated Inheritance (Switch Statement)
Several `if` statements in a row (comparing the same item)	
Use of instance	
Class	
Large number of instance variable in `class`	Large Class
Large number of methods in class	
Large number of lines in class	
Complexity	
Variable, parameter, field, code fragment, method, or class never referenced	Dead Code (may also be Speculative Generality)
Code more general/complicated than it needs to be	Speculative Generality
Data	
Public fields (or just setters and getters), but little behavior in class	Data Class
Same 2-3 items occur together in clases and parameter lists	Data Clump
Fields named with similar substrings	
Alternating declaration of fields and methods	
Primitive types (int, string, etc.)	Primitive Obsession
Constants and enumerations	
Strings constants for field names	
Fields are sometimes set, sometimes null	Temporary Field
Inheritance	
Subclass is too tied to parent's data or methods	Inappropriate Intimacy (Subclass Form)
Class has little code in it	Lazy Class
Subclass throws exception on method the parent requires	Refused Bequest
Inherited method doesn't work	

Clients refer to subclass but never hold reference to the parent class	
Subclass isn't (conceptually, behaviorally) an example of parent class	
Responsibility	
Class manipulates another class's data	Feature Envy
Class relies too much on how another class works	Inappropriate Intimacy (General Form)
Chain of calls: a.b().c().d()	Message Chain
Delegation: `f() { delegate.f(); }`	Middle Man
Accommodating Change	
Introducing new class requires multiple classes at various points in its hierarchy	Combinatorial Explosion
Each level of hierarchy deals with a different attribute	
Same class changes for different reasons	Divergent Change
Adding a subclass in one hierarchy requires corresponding subclasses in other hierarchies	Parallel Inheritance Hierarchies
Subclasses in two separate hierarchies use the same prefixes	
Multiple classes must change for a single decision	Shotgun Surgery
Working with Libraries	
Library doesn't have a feature you need	Incomplete Library Class

Source: Refactoring Workbook, William C. Wake, 2004