**Code smells and anti-patterns**

|  |  |
| --- | --- |
| God class | Class that does too much |
| Long methods | Method that does too much (~5 lines) |
| And/or in name | Sign that method does more than one thing |
| Vague method/class names | Use descriptive names |
| Conditional complexity | Multiple conditionals |
| Similar switch statements |  |
| Long parameter list |  |
| Out parameters |  |
| High cyclomatic complexity | High number of discrete paths through the code |
| Abbreviated identifiers |  |
| Similar “shape” | Just squint your eyes |
| Primitive Obsession | Overuse of primitive types |
| Inappropriate intimacy | Classes that spend too much time together |
| Indecent exposure | Classes that unnecessarily expose their internals |
| Feature envy | Methods that make extensive use of another class |
| Arrow code | Heavily indented in arrow shape |
| Middle man | Class is delegating its work |
| Comments |  |
| Zombie Code | Commented out code |
| Dead code | Code that is never called |
| Service locator |  |
| Constructor over injection | Honor SRP |
| Bastard injection | Overloading constructor to provide testability |
| Magnet class | Ambiguous name that attracts |
| Magic numbers | Refactor to well named constant |
| Exceptions as control flow | Includes swallowing exceptions |
| Negative conditionals/names | Positive conditionals are easier to comprehend |
| Leaky abstractions | Concerns from one layer leaking into another |
| Side Effects | Calling method causes other unrelated consequences |
| Solution Sprawl | Five classes to do anything, consider simplifying. |
| Magic Strings | Using strings for conditionals instead of strong types, consider Enums. |
| Overuse of dynamics | Increases complexity. |