Code Smells

Thierry Sans

Bloaters

- **Long Method**
  More than 10 lines should raise questions
- **Large Class**
  Too many fields/methods/lines of code
- **Long Parameters List**
  More than 3 or 4 parameters for a method should raise questions
- **Data Clumps**
  Pieces of code that contain the same identical group of variables

Object-Oriented Abusers

- **Conditional Complexity**
  Complex switch operator or sequence of if statements
- **Temporary Field**
  Classes that contains optional and underused attributes
- **Refused Bequest**
  Inheriting from a class, but never use any of the inherited functionality
- **Alternative Classes with Different Interfaces**
  Two classes perform identical functions but have different method names

Change Preventers

- **Divergent Change**
  Changing a class requires you change many unrelated methods
- **Shotgun Surgery**
  Making any modification requires that you make many small changes to many different classes
- **Parallel Inheritance Hierarchies**
  Create a subclass for a class requires you to create a subclass for another class

Dispensables

- **Comments**
  A method is filled with explanatory comments
- **Duplicate Code**
  Two code fragments look <u>almost</u> identical
- **Lazy Class**
  Maintaining a class that dos not do much
- **Data Class**
  A data class refers to a class that contains only fields and getters and setters
- **Dead Code**
  A variable, parameter, field, method or class is no longer used
- **Speculative Generality**
  There is an unused class, method, field or parameter

Couplers

- **Feature Envy**
  A method accesses the data of another object more than its own data
- **Inappropriate Intimacy**
  One class uses the internal fields and methods of another class
- **Message Chains**
  In code you see a series of calls resembling $a->b()->c()->d()
- **Middle Man**
  If a class performs only one action, delegating work to another class