

Pattern	Description	Comments
Use Streams	Let strings and streams play nice together. Use <code>String&gt;&gt;streamContents: [aBlock]</code>	Don't be overly concerned with efficiency, but repeated string concatenation is a misuse of the libraries.
Don't repeat yourself (DRY)	Everything should be said once, but only once.	
Don't Plan ahead	Don't put in hooks for functionality that you may not need	If you DRY, then it should be easy to add the functionality <i>if and when</i> you do need it
Semantically meaningful names	Names should be meaningful to readers, not just to you.	Skublics Ch 1 is all about this. It's probably the most powerful hint for more maintainable code.
Typed names when appropriate	Use names like "anInteger" or "an Object" only when that is the best choice.	A "semantic" name may be better. See Skublics Ch 1 again!
Use self changed: anAspect protocol	Make sure that <i>self changed:</i> is sent <b>whenever</b> something significant changes	Use accessor methods to set your inst vars. If the accessor sends <i>self changed:</i> , it can't be "forgotten"
No dead code	dead code and dead variables should be deleted (not just commented out)	
Rule of Demeter	A chain of sends like <i>dieView bounds width</i> exposes too much about the internals of <i>dieView</i> . Add a method <i>dieView lenthOfSide</i>	
Use <i>extract method</i> refactoring	... when there is a related group of statements that could benefit from being named and grouped.	
Don't use isKindOf:	It's usually bad to check the class of an object, both to report errors and to write conditional code.	Are tests an exception?
Observe the view-model separation	<i>If</i> you decide that it's a good idea to separate the view from the model, <i>then</i> follow through	It should at least be possible to create a model object without also creating a view on it!
Magic Numbers	Don't pepper your code with magic numbers. Any number that appears more than once should be turned into a (constant) method.	Is that number really an independent constant? Or should the method calculate it from another feature?

Pattern	Description	Comments
Use explaining temporary variables	Assign a complicated expression to a temporary variable whose name explains the role of that expression	If the expression involves more than a simple variable access (e.g., <i>aBag sortedCounts</i> ), you should not repeat it, but should cache the result in a temp.
Don't use temps	Sometimes, code is shorter and clearer if you don't use temps at all but program functionally.	The message <i>yourself</i> exists to make this style easier.
Name your protocols	If your methods are all " <i>not yet classified</i> ", take the hint.	Follow the conventions used in the rest of the system.