

Contents

Acknowledgements	9
Preface	11
1 Birth	13
1.1 Never use -er names	15
1.2 Make one ctor primary	22
1.3 Keep ctors code-free	27
2 School	35
2.1 Encapsulate as little as possible	36
2.2 Encapsulate something at the very least	40
2.3 Always use interfaces	43
2.4 Choose method names carefully	46
2.4.1 Builders are nouns	47
2.4.2 Manipulators are verbs	50
2.4.3 Examples of both	53
2.4.4 Boolean results	54
2.5 Don't use public constants	57
2.5.1 Introduction of coupling	59
2.5.2 Loss of cohesion	61
2.6 Be immutable	65
2.6.1 Identity mutability	68
2.6.2 Failure atomicity	69

2.6.3	Temporal coupling	71
2.6.4	Side effect-free	73
2.6.5	No NULL references	74
2.6.6	Thread safety	76
2.6.7	Smaller and simpler objects	80
2.7	Write tests instead of documentation	82
2.8	Don't mock; use fakes	85
2.9	Keep interfaces short; use smarts	93
3	Employment	95
3.1	Expose fewer than five public methods	97
3.2	Don't use static methods	99
3.2.1	Object vs. computer thinking	101
3.2.2	Declarative vs. imperative style	105
3.2.3	Utility classes	113
3.2.4	Singleton Pattern	115
3.2.5	Functional programming	120
3.2.6	Composable decorators	122
3.3	Never accept NULL arguments	126
3.4	Be loyal and immutable, or constant	133
3.5	Never use getters and setters	145
3.5.1	Objects vs. data structures	146
3.5.2	Good intentions, bad outcome	150
3.5.3	It's all about prefixes	152
3.6	Don't use <code>new</code> outside of secondary ctors	154
3.7	Avoid type introspection and casting	159
4	Retirement	165
4.1	Never return NULL	167
4.1.1	Fail fast vs. fail safe	171
4.1.2	Alternatives to NULL	172
4.2	Throw only checked exceptions	176

4.2.1	Don't catch unless you have to	179
4.2.2	Always chain exceptions	182
4.2.3	Recover only once	185
4.2.4	Use aspect-oriented programming	187
4.2.5	Just one exception type is enough	189
4.3	Be either final or abstract	190
4.4	Use RAII	197
Epilogue		199
Bibliography		201
Index		203