List of Soundbites

Page 9 *When you find you have to add a feature to a program, and the program's code is not structured in a convenient way to add the feature; first refactor the program to make it easy to add the feature, then add the feature.*

Page 10 *Before you start refactoring, check that you have a solid suite of tests. These tests must be self-checking.*

Page 14 *Refactoring changes the programs in small steps: so if you make a mistake, it is easy to find where the bug is.*

Page 17 *Any fool can write code that a computer can understand, good programmers write code that humans can understand.*

Page 57 *Refactoring is a change made to the internal structure of a software component to make it easier to understand and cheaper to modify, without changing the observable behavior of that software component.*

Page 65 *Don't publish interfaces prematurely. Modify your code ownership policies to smooth refactoring.*

Page 72 *Surely if two programmers are working together on the same machine they can only go half as fast two programmers working separately? This would be true if the hardest part of programming was typing.*

Page 83 *"If it stinks, change it."*
— Grandma Beck, discussing child raising philosophy

Page 93 *When you feel the need to write a comment, try first to refactor the code so that any comment would be superfluous.*

Page 96 *Make sure all tests are fully automatic and check their own results.*

Page 96 *A suite of tests is a powerful bug detector that decapitates the time it takes to find bugs.*

Page 97 *Don't let the fear that tests can't catch every bug stop you from testing. If your tests only get half the bugs, they are still worthwhile, and you will usually do much better than that.*

Page 101 *Run your tests frequently. Localized tests whenever you compile, every test at least every day.*

Page 106 *Think of the boundary conditions where things that might go wrong and concentrate your tests there*

Page 107 *Don't forget to test that exceptions are raised when things should go wrong*

Page 381 *Make a clear distinction between Reference Objects (e.g. Person) and Value Objects (e.g. Date). Value objects should always be immutable.*

Page 382 *Refactoring is an active form of code inspection.*

Page 406 *You do the early refactorings to learn more about the program, these set you up for later ones that really simplify the structure.*

Page 408 *Don't let fear of the future stop you from doing a refactoring now. Refactorings aren't difficult to change, and learning you gain repays the effort.*