Programmer's Dozen

Thirteen Recommendations for Reviewing, Refactoring and Regaining Control of Code

When you look for it, there appears to be no shortage of technical wisdom on how to develop clear and robust code. However, the quantity and consistency of such wisdom can often leave programmers swamped. Ideally there should be common themes that can be found across the many different specific practices identified as great and good.

This seminar offers a concrete thirteen-point list of practices (zero through twelve) that can be applied out-of-the-box to reduce code size and complexity, acting as both guidelines for new code and indicators for refactoring. The short list has no ambition to be all that you needed to know about design but were afraid to ask, but it does offer an easily memorable and easily practised set of guidelines with a great immediate return on investment for any programmer, project or company.

Audience

The seminar is suitable for programmers familiar with object orientation and modern programming languages, such as Java, C++ and C#.

Content

Introduction Code and the role of design · Losing control of system development · Minimalism · The practice of refactoring · Recommendations for new and existing code

Recommendation 0 ***Prefer Code to Comments*** Comments as information versus distraction · Comments that add no value · Comments that have negative value

Recommendation 1 ***Follow Consistent Form*** Idioms versus idiolects · Expectation · Value conversions · Overloading · Operator overloading · The case for inconsistency

Recommendation 2 ***Employ the Contract Metaphor*** Interfaces as contracts · Quality of failure · When assertiveness is not enough · Substitutability and inheritance · Weak contracts

Recommendation 3 ***Express Independent Ideas Independently*** Interface decoupling · Layering · Noosely coupled code · Separating unrelated command and query operations

Recommendation 4 ***Encapsulate*** The deeper meaning of encapsulation · Affordances · Whole values · Combining sequentially dependent operations · Sealing porous layers

Recommendation 5 ***Parameterize from Above*** Problems with singleton, global and class static objects · Testability · Inverting dependencies · Mock objects · Decoupled concurrency

Recommendation 6 ***Restrict Mutability of State*** State, changes and bugs · Immutable values · Copying · Qualification for restricting operations

Recommendation 7 ***Favour Symmetry over Asymmetry*** Symmetry and simplicity · The role of asymmetry · Resource management · Balanced interfaces · The temptations of symmetry

Recommendation 8 ***Sharpen Fuzzy Logic*** The cost of confused logic · Logical relationships · Short circuiting · Existential indirection · The (lack of) meaning of Boolean arguments

Recommendation 9 ***Go with the Flow*** Linear versus block-structured code · Turbulent versus smooth control flow · Taking similar decisions at the same level · Exceptions are exceptional

Recommendation 10 ***Let the Code Take some Decisions*** Explicit versus emergent conditional logic · Duplicate decisions · Collective decisions · Polymorphism · Lookup tables

Recommendation 11 ***Omit Needless Code*** Code as less than the sum of its parts · Unhelpful, useless and unreachable code · Optimisations and conveniences that aren't

Recommendation 12 ***Unify Duplicate Code*** Forms of duplication · Extracting operations · Extracting classes · A visit to the library

Outroduction Summary · Conclusions

Additional Details

Duration	1 day
Setup	Projection facilities for a PC · Whiteboards and/or flip charts
Contact	Kevlin Henney · kevlin@curbralan.com · Curbralan Limited · +44 117 942 2990