



Refactoring – What Is It?

- Mhat is refactoring?
 - m Modifying code to improve its structure without changing functionality
 - "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure" (Fowler)
- Mhat is the opposite of refactoring?
- Mhy might one want to do it?



Refactoring – Why Do It?

Why is it necessary?

- A long-term investment in the quality of the code and its structure
 - Without proper maintenance, code tends to "rot" as its structure deteriorates when quick last-minute fixes are made and unplanned features are added
- Doing no refactoring may save on costs in the short term but pays huge interest in the long run "Don't be penny-wise but hour-foolish!"

Why fix it if it ain't broken? Every module has three functions:

- (a) to execute according to its purpose;
- (b) to afford change;(c) to communicate to its readers.
- It it doesn't do one or more of these, it's broken.



Refactoring – When to Do It?

Refactoring is necessary from a business standpoint too

- Helps with predictable schedules and high output at lower cost
- ROI for improved software practices is 500% (!) or
- By doing refactoring a team saves on unplanned defect-correction work

When is refactoring necessary?

- Best done continuously, along with coding and
- Very hard to do late, much like testing
- n Often done before plunging into version 2



Types of Refactoring and Reasons for Doing It



Types of Refactoring

- Renaming (methods, variables) Naming (extracting) "magic" constants
 - Extracting common functionality into a service / module / class / method
 - Extracting code into a method
 - Changing method signatures
 - Splitting one method into several to improve cohesion and readability (by reducing its size)
 - Putting statements that semantically belong together near each other
 - Exchanging risky language idioms with safer alternatives
 - Clarifying a statement (that has evolved over time or that is hard to "decipher")
 - Performance optimization

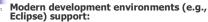


Summary: Top Reasons for Refactoring

- Improving readability (and hence productivity)
- Responding to a change in the spec/design by improving cohesion
 - n Or anticipating such a change
- "If bug rates are to be reduced, each function needs to have one well-defined purpose, to have explicit singlepurpose inputs and outputs, to be readable at the point where it is called, and ideally never return an error condition." Steve Maguire -- "Writing Solid Code"



Language Support for Refactoring



- variable/method/class renaming
- method or constant extraction
- extraction of redundant code snippets
- method signature change
- extraction of an interface from a type
- method inlining
- providing warnings about method invocations with inconsistent parameters
- help with self-documenting code through auto-completion
- Older development environments (e.g., vi, Emacs, etc.) have little or no automated support
- Discourages programmers from refactoring their code



Your Questions on Refactoring



Main Take-Away Points on Refactoring