Refactoring, A Whole-Team Guide

Refactoring is…? Changing the design of existing code without changing its observable behavior (per Martin Fowler).

These changes include renaming, reducing duplication, rearranging code, and more. Even large refactorings are built from a series of small, safe transformations.

Is Refactoring Rework? Not usually. Refactoring shifts some design work to later in the process (when the programmer has more information). Refactoring also helps us implement complex code by starting with simpler code and evolving it.

Benefits:
* Code is smaller and easier to understand.
* The design is easier to extend to support new capabilities.
* Less duplication means less chance of coordinated code getting out of sync.
* Improving how the parts connect makes future changes affect fewer places.

Risks:
* **The biggest risk: programmers say "refactor" when they really mean "rewrite".** *Refactoring* creates code that is provably equivalent; *rewriting* is re-doing a section of code. Both are valid, but refactoring is much safer.
* Refactoring is a skill that must be learned.
* Any change carries some risk, but we try to make refactoring low-risk. All of these are valid ways to refactor, but some are safer than others:

Safer	Less Safe
Automated (tool-supported)	Manual
Manual, following a defined recipe	Ad-hoc manual
Covered by tests	Not covered by tests
Small refactorings	Big refactorings

* Other processes and tools can reduce risk too, such as code review (by pair, mob, or another person), and using a version-control system (e.g., git).
* Refactoring tools work within a programming language. But systems often have parts that are outside the programming language. For example, a configuration file may need updating when a name changes.

How Do You Manage Refactoring? Refactoring is mostly an integrated part of programming work, not a separable activity.

The exception is that a team may (rarely) want a large refactoring to be done all at once. (Large refactorings can usually be interleaved with other work.) This may be something that can be deferred, but gets more expensive over time. The programmers had best consult with others before taking that path. If this happens more than "very rarely", you should explore why.

The full article is at https://xp123.com/refactoring-whole-team-guide/