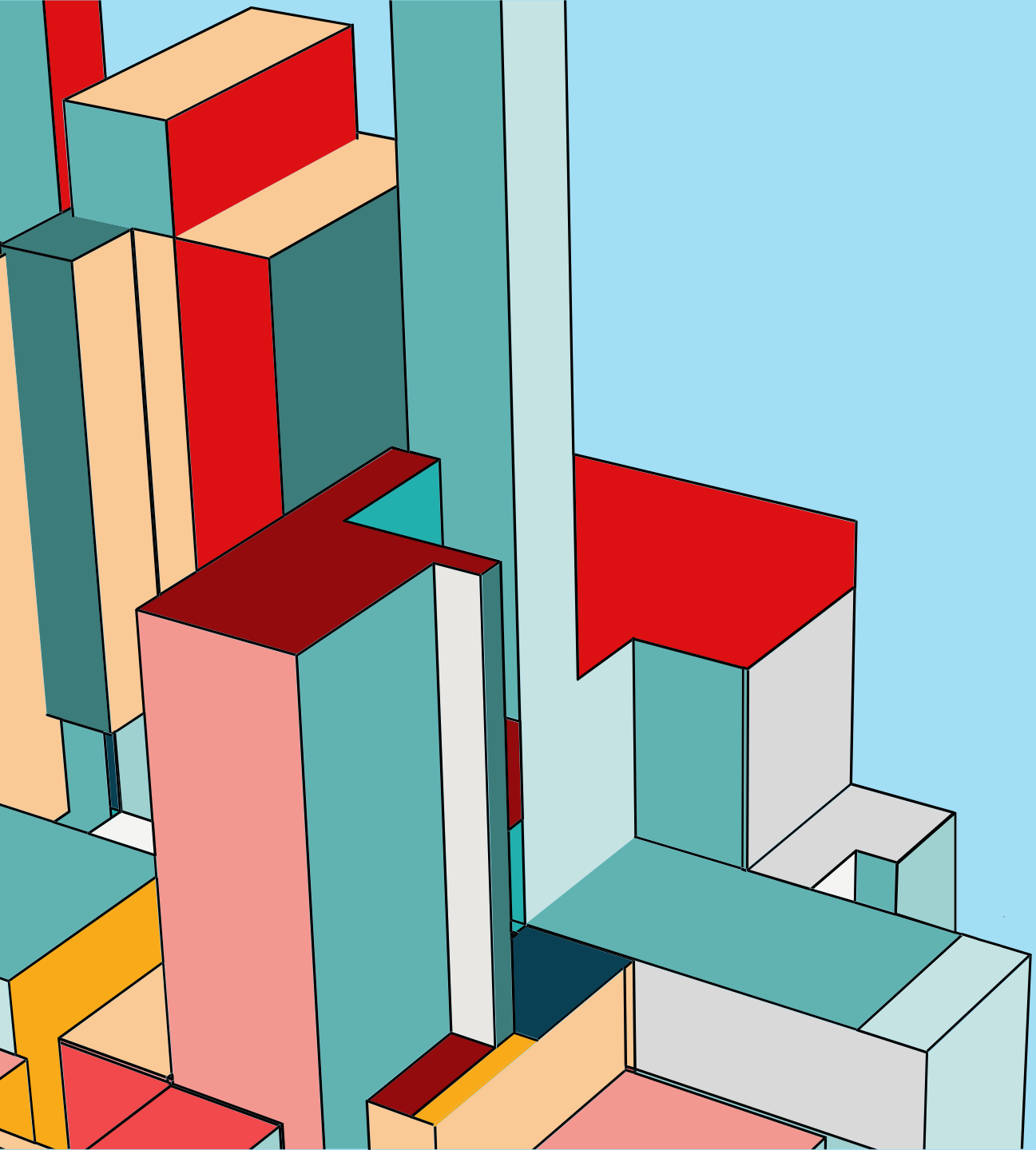
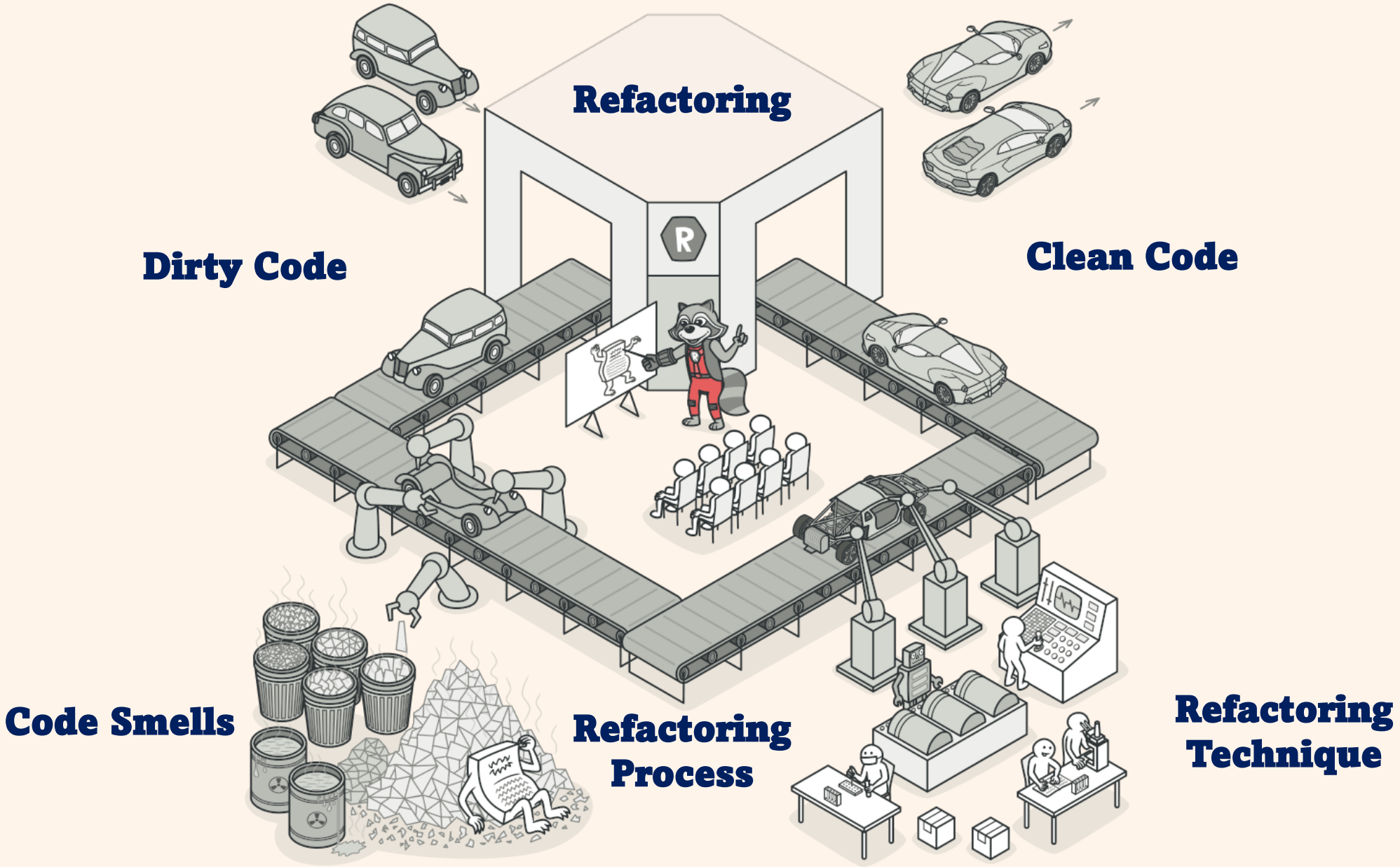
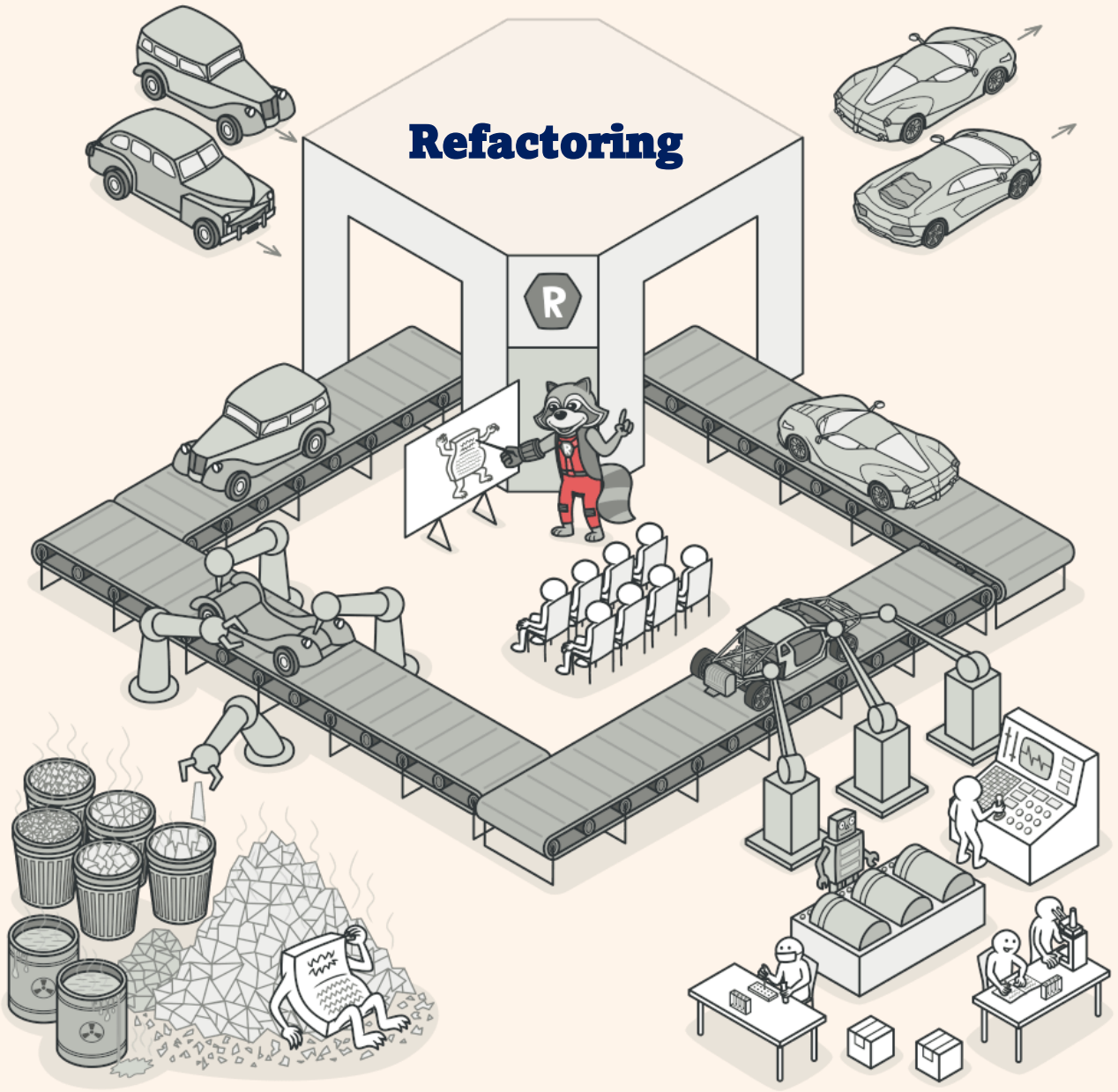


REFACTORING & DESIGN PATTERNS PART 1

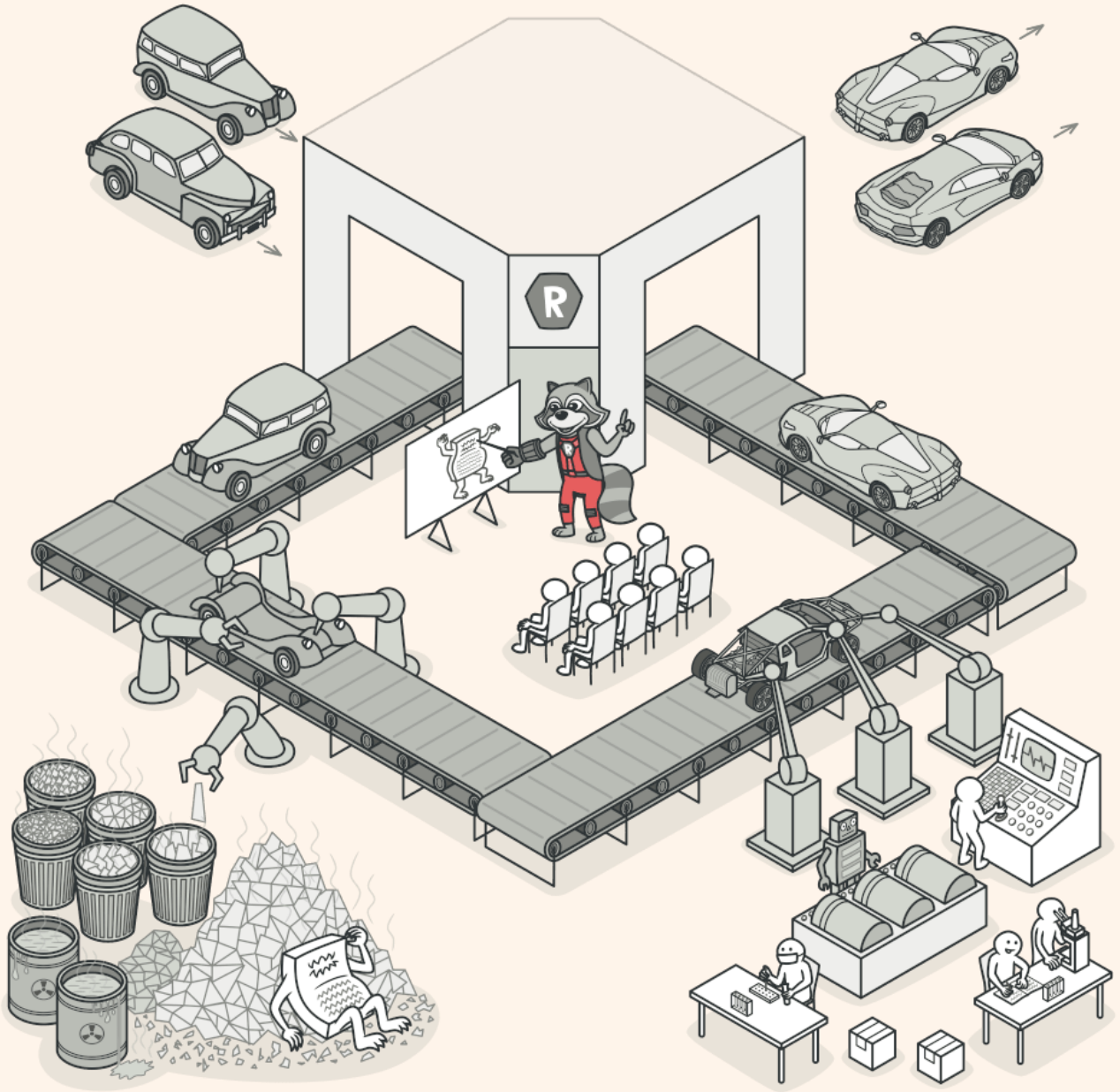

Refactoring
Dirty Code
Clean Code
Code Smells
Refactoring Process
Refactoring Technique

Refactoring is a systematic process of improving code without creating new functionality that can transform a mess into clean code and simple design.



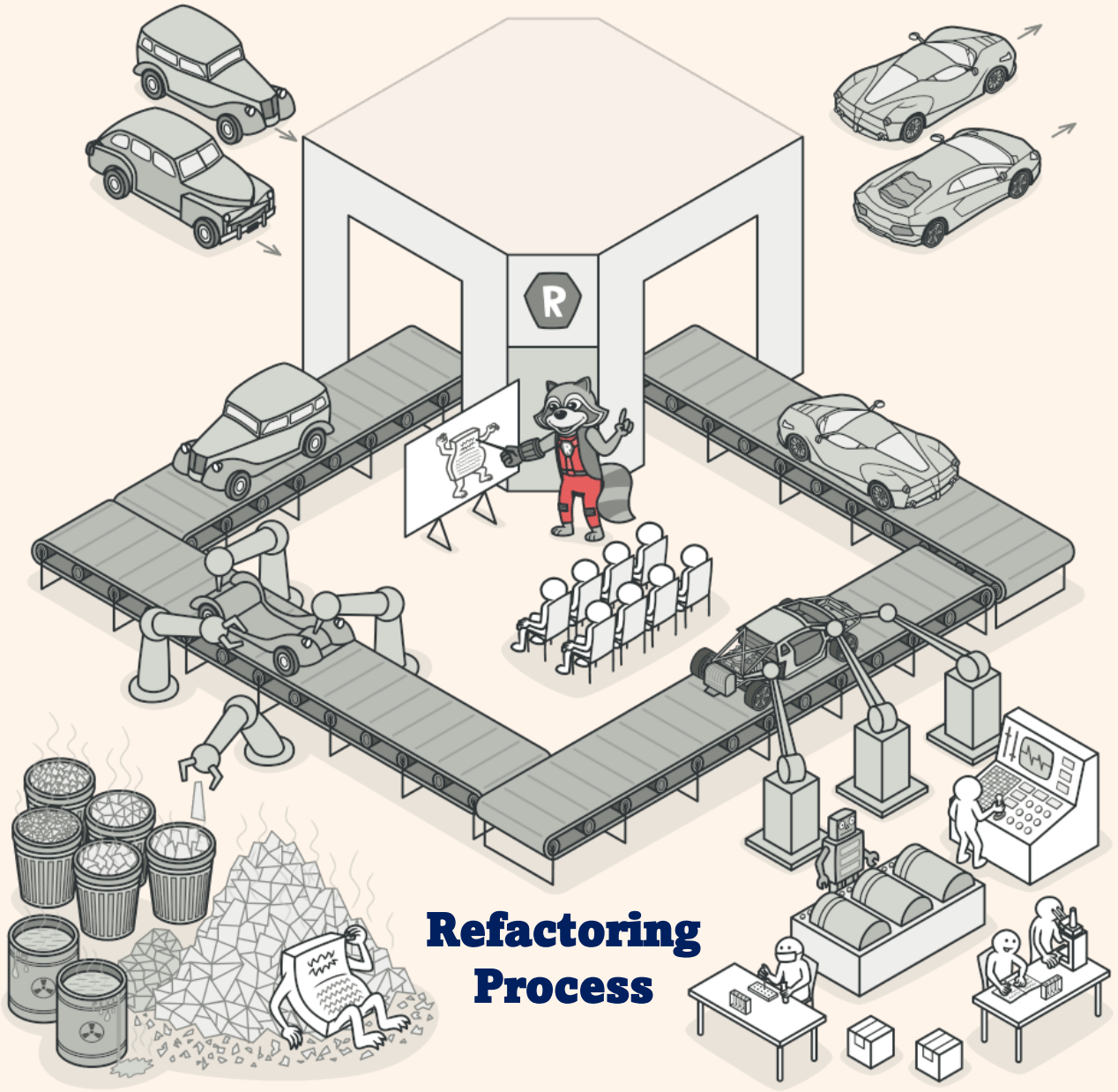
Dirty code is result of inexperience multiplied by tight deadlines, mismanagement, and nasty shortcuts taken during the development process.

Code smells are indicators of problems that can be addressed during refactoring. Code smells are easy to spot and fix, but they may be just symptoms of a deeper problem with code.



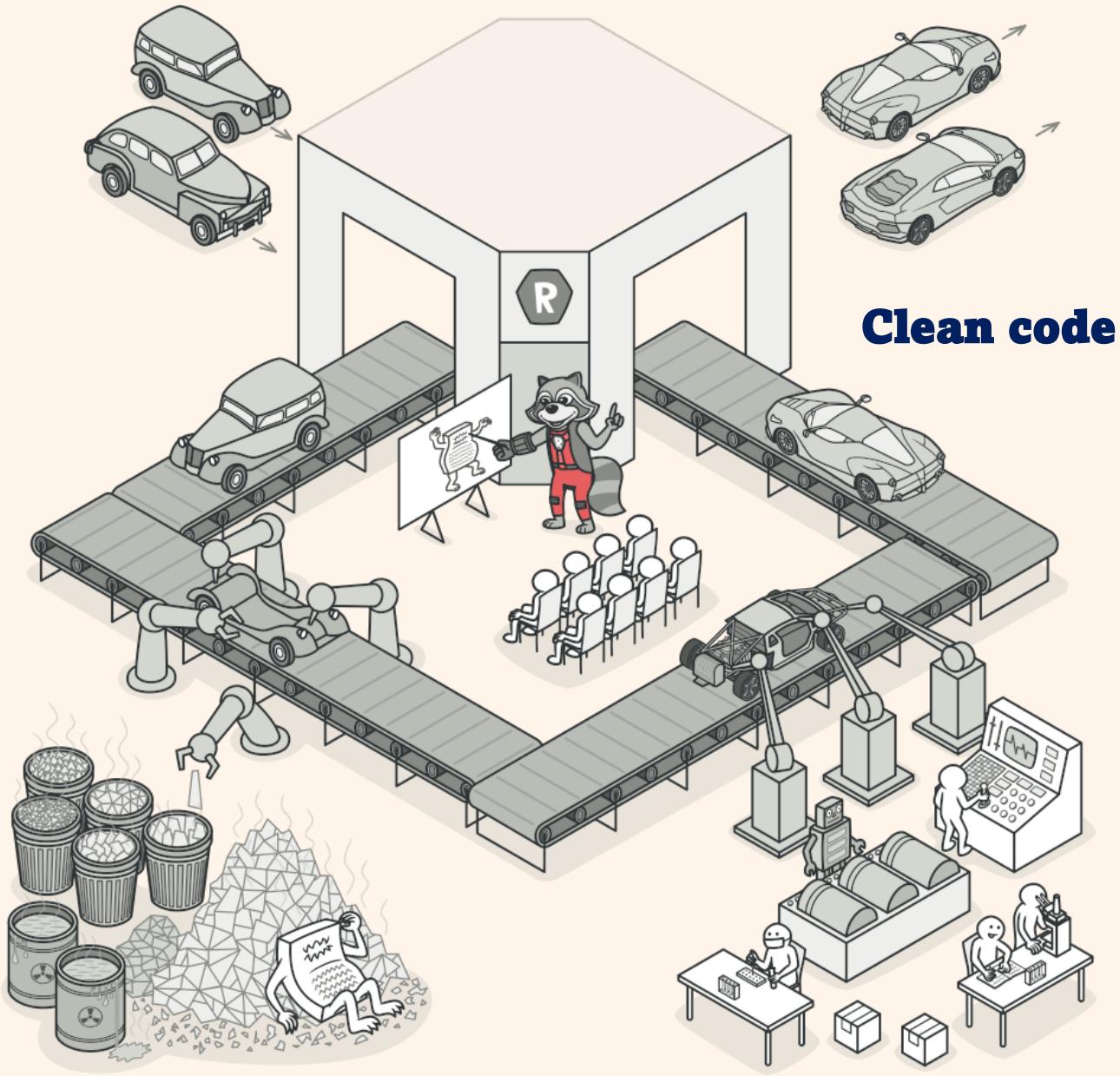
Code Smells

Performing refactoring step-by-step and running tests after each change are key elements of refactoring that make it predictable and safe.

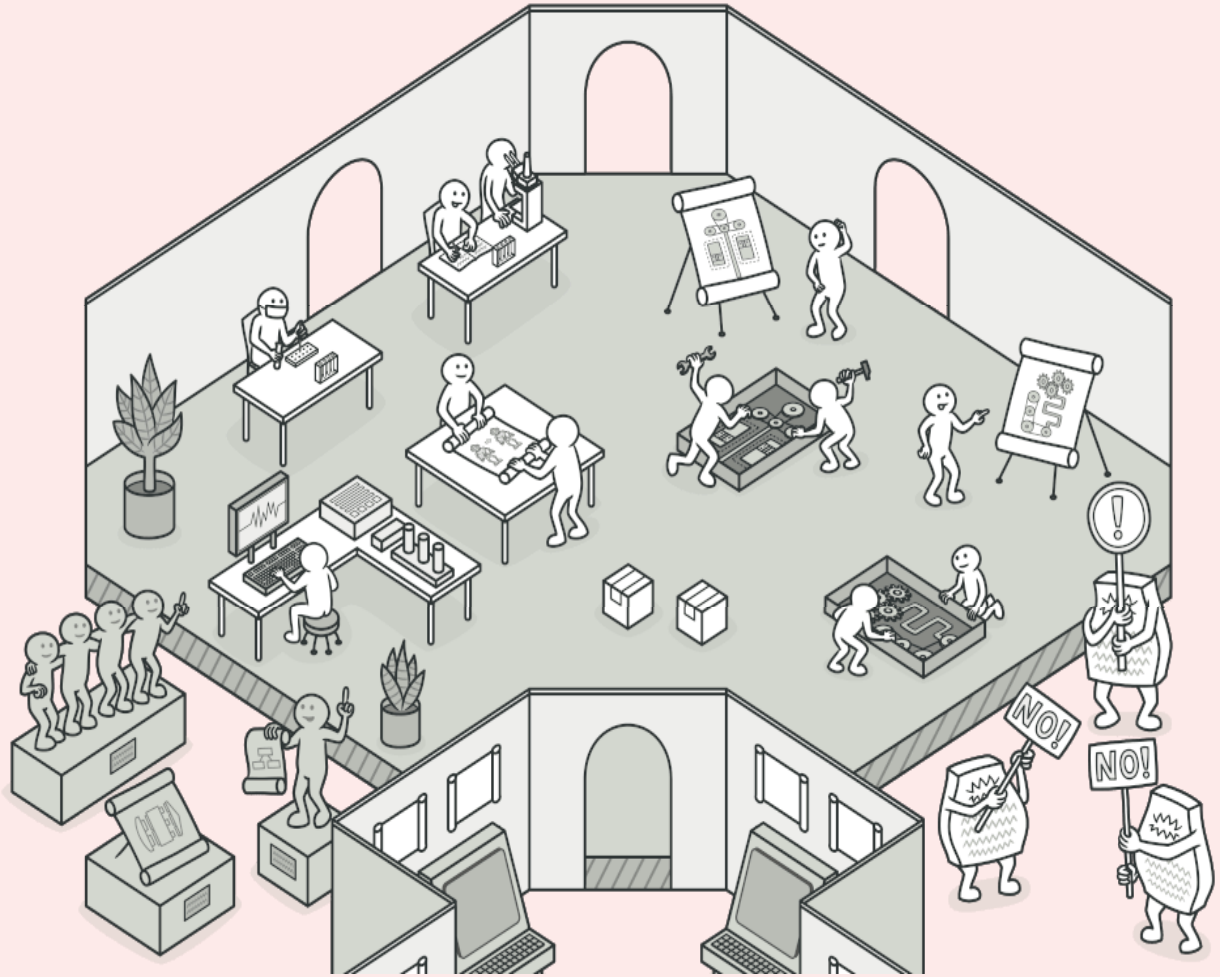


Actual refactoring steps. Most refactoring techniques have their pros and cons. Therefore, each refactoring should be properly motivated and applied with caution.

Clean code is code that is easy to read, understand and maintain. Clean code makes software development predictable and increases the quality of a resulting product

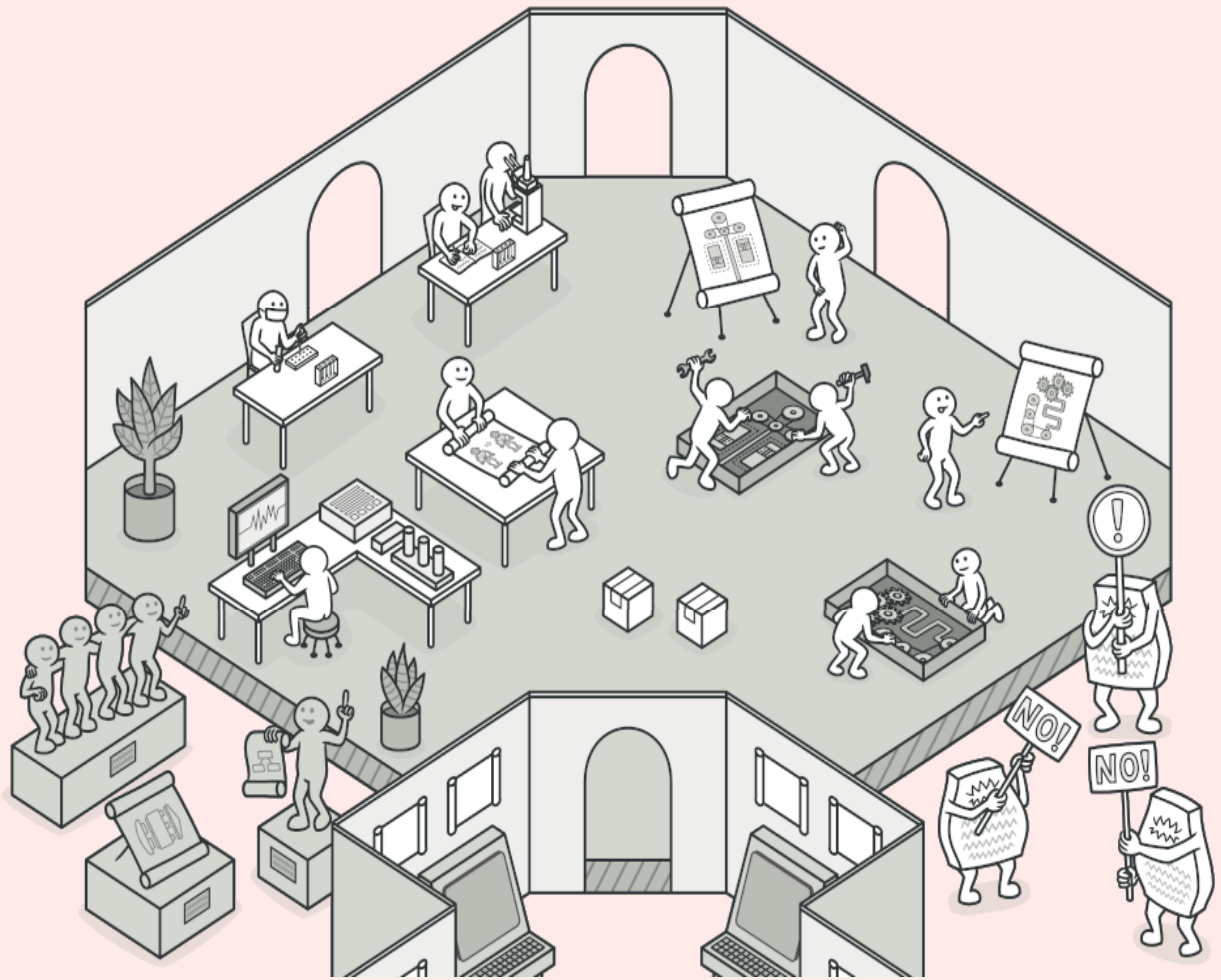


DESIGN PATTERNS



DESIGN PATTERNS

Design patterns are typical solutions to common problems in software design. Each pattern is like a blueprint that you can customize to solve a particular design problem in your code.



WHAT DOES THE PATTERN CONSIST OF?

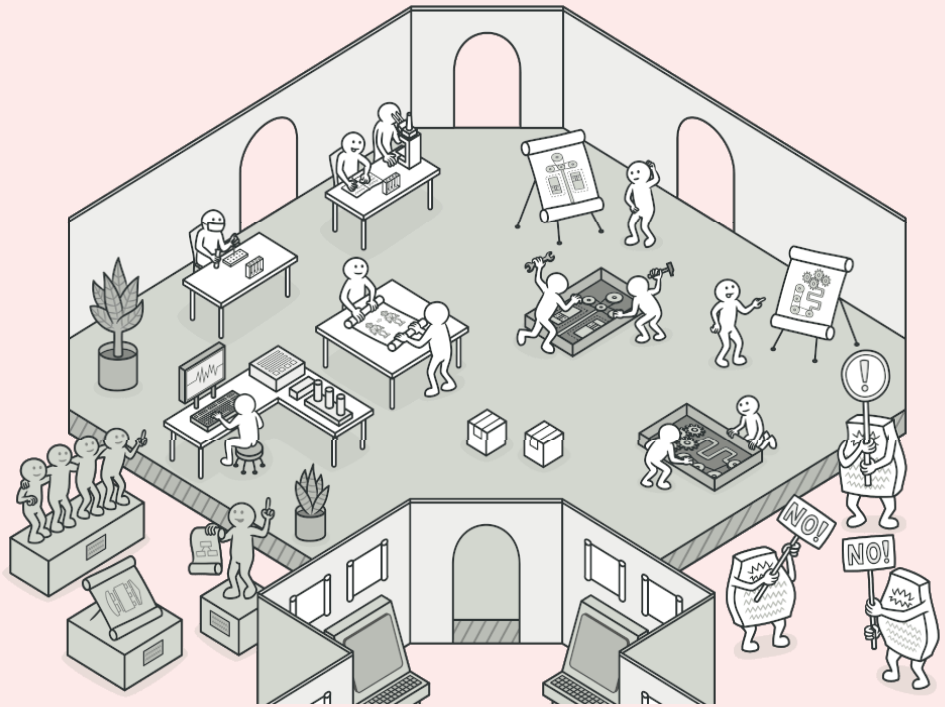
Most patterns are described very formally so people can reproduce them in many contexts. Here are the sections that are usually present in a pattern description:

Intent of the pattern briefly describes both the problem and the solution.

Motivation further explains the problem and the solution the pattern makes possible.

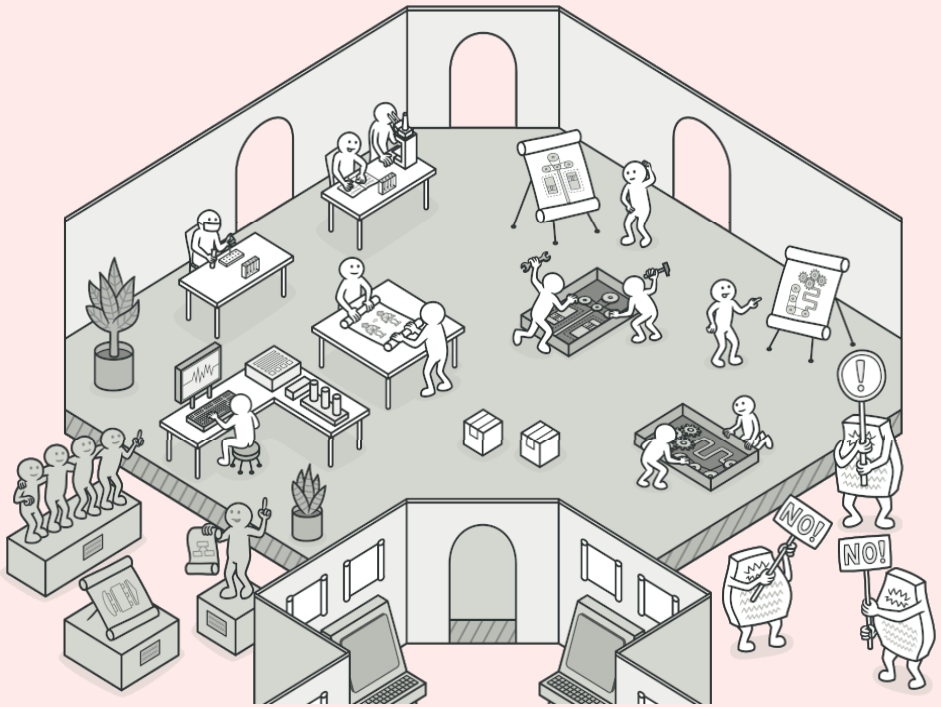
Structure of classes shows each part of the pattern and how they are related.

Code example in one of the popular programming languages makes it easier to grasp the idea behind the pattern.



HISTORY OF PATTERNS

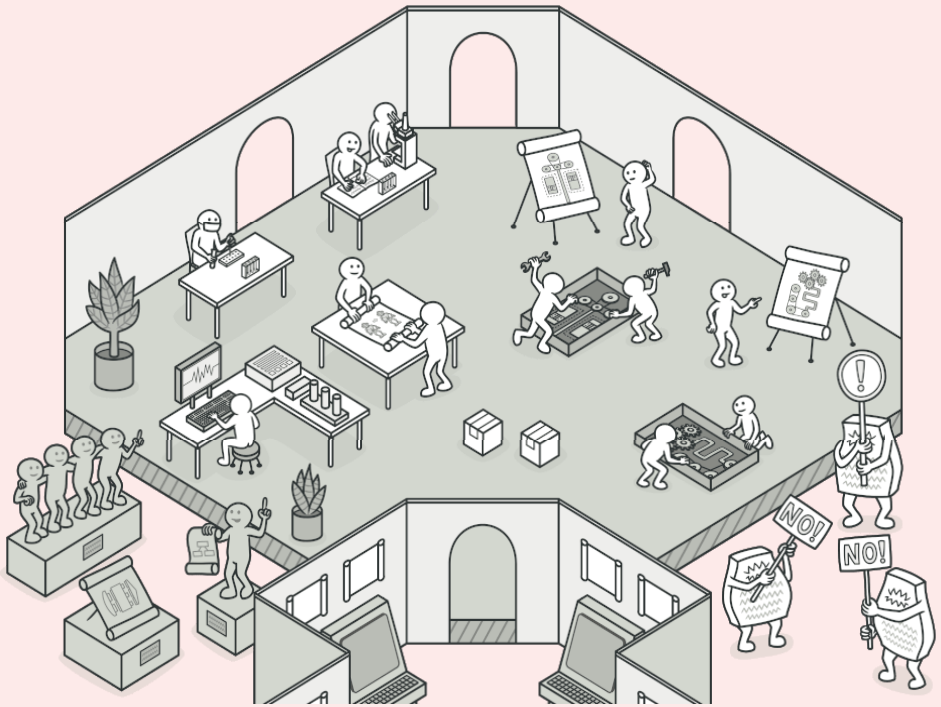
The concept of patterns was first described by Christopher Alexander in A Pattern Language: Towns, Buildings, Construction. The book describes a “language” for designing the urban environment. The units of this language are patterns. They may describe how high windows should be, how many levels a building should have, how large green areas in a neighborhood are supposed to be, and so on.



HISTORY OF PATTERNS

The idea was picked up by four authors: Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm. In 1994, they published Design Patterns: Elements of Reusable Object-Oriented Software, in which they applied the concept of design patterns to programming. The book featured 23 patterns solving various problems of object-oriented design and became a best-seller very quickly. Due to its lengthy name, people started to call it “the book by the gang of four” which was soon shortened to simply “the GoF book”.

WHY SHOULD I LEARN PATTERNS?



Design patterns are a toolkit of tried and tested solutions to common problems in software design. Even if you never encounter these problems, knowing patterns is still useful because it teaches you how to solve all sorts of problems using principles of object-oriented design.