Clean Code: Writing Maintainable Software

Master the art of writing code that's easy to read, fix, and extend. For software developers and engineers.

Programming Instructor: David Clarke • 5 sections • 50 lessons • 16.00h total

View this course online Browse all courses

About this course

Many developers face the challenge of code that becomes difficult to maintain over time, leading to increased bugs and slower development cycles. This course tackles this problem by teaching the fundamental principles of clean code.

You'll learn practical techniques for naming, function design, refactoring, and applying design patterns through video lessons, real-world examples, and hands-on exercises. By the end, you'll be able to write code that is both functional and easy to maintain.

Course curriculum

Foundations of Clean Code

- What is Clean Code and Why It Matters (12m)
  We'll define clean code and discuss how it impacts software longevity and team productivity.
- Setting Up Your Coding Environment for Quality (8m)
  Tools and settings that help you write better code from the start.
- Naming Variables and Functions Clearly (18m)
  Step-by-step guide to choosing names that convey intent without confusion.
- The Art of Writing Simple Functions (25m)
  How to break down complex tasks into small, manageable functions.
- Effective Use of Comments and Documentation (12m)
  When to comment and how to write documentation that actually helps.
- Basic Error Handling for Robust Code (18m)
  Techniques to handle errors gracefully without cluttering your code.
- Introduction to Code Formatting Standards (8m)
  Why consistent formatting matters and popular style guides.
- Avoiding Common Pitfalls in Early Development (15m)
  Real case study on mistakes that lead to unmaintainable code.

Core Concepts for Readable Code

- Deep Dive into Code Duplication (22m)
  Identifying and eliminating duplication to reduce bugs and effort.
- SOLID Principles Overview (20m)
  An introduction to the five principles that guide object-oriented design.
- Single Responsibility Principle in Practice (18m)
  How to ensure each class or module has one clear purpose.
- Open/Closed Principle for Extensible Code (15m)
  Writing code that's open for extension but closed for modification.
- Liskov Substitution and Interface Segregation (25m)
  Understanding and applying these principles to avoid design flaws.
- Dependency Injection for Loose Coupling (20m)
  Managing dependencies to make code easier to test and change.
- Testing Strategies for Maintainable Code (30m)
  Writing tests that catch issues without being a burden to maintain.
- Refactoring Techniques: Extract and Rename (18m)
  Practical refactoring methods to improve code structure step by step.
- Code Smells and How to Address Them (22m)
  Recognizing signs of poor code and fixing them systematically.
- Introduction to Design Patterns for Clean Code (25m)
  Common patterns that help create maintainable and scalable software.

Deep Dive into Code Quality

- Advanced Naming: Context and Conventions (15m)
  Tailoring names to domain-specific contexts for better clarity.
- Decomposing Complex Functions Effectively (20m)
  Strategies for handling large functions and improving readability.
- Managing State and Side Effects (22m)
  How to control state changes to avoid unexpected behaviors.
- Balancing Performance and Readability (18m)
  When and how to optimize code without sacrificing maintainability.
- Conducting Effective Code Reviews (12m)
  Best practices for giving and receiving feedback on code quality.
- Using Static Analysis Tools (25m)
  Tools like linters and formatters that enforce clean code standards.
- Integrating Code Quality into CI/CD Pipelines (20m)
  Automating checks to maintain code quality throughout development.
- Real-World Case Study: Improving a Legacy System (30m)
  Analyzing how a team refactored a messy codebase into clean code.

Advanced Application and Patterns

- Applying SOLID in Microservices Architecture (25m)
  How clean code principles adapt to distributed systems.
- Designing APIs for Long-Term Maintainability (20m)
  Creating interfaces that are easy to use and evolve over time.
- Clean Database Code and ORM Practices (18m)
  Writing efficient and readable database interactions.
- Refactoring Legacy Code: A Step-by-Step Guide (32m)
  Techniques to safely modernize old code without breaking it.
- Team Practices for Sustaining Clean Code (15m)
  Establishing coding standards and reviews in a team setting.
- Case Study: Clean Code in a High-Traffic Application (22m)
  How a major app maintains code quality under pressure.
- Advanced Design Patterns: Strategy and Observer (20m)
  Using these patterns to write flexible and clean code.
- Code Metrics and How to Use Them (18m)
  Measuring code quality with metrics like complexity and coverage.
- Handling Technical Debt Proactively (15m)
  Strategies to identify and reduce debt before it becomes critical.
- Clean Code in DevOps and Infrastructure (20m)
  Applying maintainability principles to scripts and configurations.
- Cross-Language Clean Code Principles (22m)
  Adapting clean code concepts across different programming languages.

- Building a Personal Clean Code Toolkit (18m)
  Tools and habits that help you write better code every day.

Implementation and Moving Forward

- Setting Up a Clean Code Workflow (20m)
  Integrating clean code practices into your daily development process.
- Automating Code Quality with Linters and Formatters (15m)
  Configuring tools to enforce standards automatically.
- Measuring and Improving Code Maintainability (18m)
  Using metrics to track progress and identify areas for improvement.
- Developing Personal Coding Habits for Quality (12m)
  Daily routines that help you write cleaner code consistently.
- Collaborating on Clean Code with Teams (20m)
  Communication and practices to align team efforts on code quality.
- Handling Bugs in Clean Codebases (15m)
  Debugging techniques that leverage clean code structure.
- Future-Proofing Your Code for 2026 and Beyond (22m)
  Trends and practices to keep your code maintainable long-term.
- Creating a Clean Code Culture in Your Organization (25m)
  Steps to advocate for and implement clean code at a larger scale.
- Resources for Continuous Learning (10m)
  Books, websites, and communities to stay updated on clean code.
- Final Project: Refactoring a Sample Application (32m)
  Apply everything you've learned by improving a provided codebase.
- Review and Self-Assessment (15m)
  Check your understanding and identify areas for further study.
- Next Steps in Your Clean Code Journey (12m)
  How to continue practicing and integrating clean code into your work.