Publication date : 02/17/2026

Course : Clean Code

Practical course - 2d - 14h00 - Ref. CCJ
Price : 1500 CHF E.T.

NEW

This course aims to transform your approach to development around two fundamental pillars: code quality and code security. From producing clear, efficient code to structuring simple, coherent functions, you'll see how to produce code that everyone can understand.

Teaching objectives

At the end of the training, the participant will be able to:

- Assessing and reducing technical debt
- Produce readable, expressive code
- Structure simple, coherent functions
- Putting SOLID principles into practice
- Implement an effective testing strategy
- Refactoring legacy code safely
- Industrializing code quality
- Integrating AI as a development assistant

Intended audience

Object programming developers.

Prerequisites

Significant programming experience, ideally in an object-oriented language. Good command of an IDE and basic knowledge of unit testing.

Practical details

Hands-on work

Analysis, exchange, practical work.

Course schedule

PARTICIPANTS

Object programming developers.

PREREQUISITES

Significant programming experience, ideally in an object-oriented language. Good command of an IDE and basic knowledge of unit testing.

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

1 From coder to software craftsman

- Technical debt: measuring the real cost of faulty code.
- The Boy Scout rule: leave the code cleaner than you found it.
- The impact of AI: how ChatGPT and Copilot are changing the game.

Hands-on work

Collective analysis of a "dirty" code snippet. Identification of code smells and AI-assisted rewriting to compare results.

2 The art of naming and clarity

- The intention above all: to choose names that reveal the "why" and not the "how".
- Eliminate ambiguity: avoid mental mappings and unpronounceable names.
- Modern conventions: naming classes and methods in the era of modern frameworks.

Hands-on work

Refactoring of cryptic variables and functions (e.g. d, list1, proc()) into clear business concepts.

3 Narrative functions and complexity

- Single Responsibility Principle (functions): a function can only do one thing.
- Levels of abstraction: don't mix the high level (business) with the low level (technical details).
- Arguments and side effects: why avoid boolean flags and limit arguments.
- Command Query Separation (CQS): separate commands from queries.

Tutored hands-on work

Take a 100-line "God Function" and break it down into 10 atomic, readable functions.

4 Applied SOLID Architecture

- SRP (Responsabilité Unique): class and module cohesion.
- OCP (open/closed): extend without modifying (the use of polymorphism).
- LSP & ISP: Liskov substitution and segregation of interfaces to avoid "Fat Interfaces".
- DIP (dependency inversion): strong decoupling to facilitate testing.
- DRY (Don't Repeat Yourself): the difference between duplicating code and duplicating knowledge.

Hands-on work

Refactoring d'un système de notification violant l'OCP et le DIP pour le rendre extensible via injection de dépendances.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.
• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.
• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

5 Test Driven Development (TDD)

- The Red-Green-Refactor cycle: the developer's breath.
- Test quality (FIRST): Fast, Independent, Repeatable, Self-Validating, Timely.
- Outillage moderne : focus sur les standards actuels (JUnit 5, Jest, Pytest).
- Mocking versus stubbing: effective use of Mockito or Jest Mocks.

Hands-on work

en groupe : "TDD Ping-Pong" Par binôme : l'un écrit le test (Red), l'autre écrit le code (Green) puis refactorisation commune.

6 Legacy Code management (new module)

- Legacy definition: code without tests.
- Dependency breaking techniques: how to test the untestable.
- Le refactoring sécurisé : utiliser l'IDE pour refactorer sans casser. Technique du "Sprout Method/Class".

Hands-on work

"Gilded Rose Kata" : mise en situation réelle. Ajouter une feature dans un code existant complexe et non testé. Sécurisation préalable par "Golden Master Testing".

7 Clean Code at scale (tooling & AI)

- Automatic code lining (ESLint, Checkstyle).
- SonarQube/Checkmarx/SonarLint: detect technical debt and security flaws in real time.
- Code Coverage: myths and realities of code coverage.
- Use AI to generate relevant documentation (JSDoc/JavaDoc).
- Prompter efficiently with AI to generate unit tests.

Hands-on work

"Pipeline Qualité". Configuration d'un set de règles strictes dans un Linter. Analyse d'un projet via SonarLint et correction des "blocker issues".

Dates and locations

REMOTE CLASS
2026 : 15 June, 17 Sep., 7 Dec.