Stand 10.06.2026

Objektorientierte Softwareentwicklung: Spezielle Programmierprinzipien mit C# - Der Weg zum Clean Code - Live-Online-Training

**Ziele -
Ihr Nutzen**

Dieser Kurs vermittelt eine fundierte Basis für das Entwickeln von wartbarem, verständlichem und qualitativ hochwertigem C#-Code.

Sie lernen, warum Clean Code wichtig ist und wie Unit Tests als Fundament für hohe Codequalität dienen. Darauf aufbauend werden die wichtigsten Prinzipien, Regeln und Best Practices für Clean Code vorgestellt und anhand praktischer Übungen direkt angewendet und vertieft. Ein weiterer Schwerpunkt des Kurses liegt auf dem Refactoring bestehender Codebasen. Zusätzlich werden C#-spezifische Features thematisiert, die sauberen Code unterstützen.

Durch zahlreiche Übungen und Beispiele gewinnen Sie ein Gespür für Clean Code und entwickeln einen klaren Blick für Codequalität, Lesbarkeit und Nachhaltigkeit.

Teilnehmer

Software-Entwickler und Software-Architekten

Voraussetzungen

Grundlegende C#-Kenntnisse

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-OOPFC#

Präsenz-Training - Deutsch

Termin	Dauer
29.06. – 01.07.2026	3 Tage

Objektorientierte Softwareentwicklung: Spezielle Programmierprinzipien mit C# - Der Weg zum Clean Code - Live-Online-Training

Inhalt

Einführung

- Was ist Software Engineering?
- Warum ist guter Code wichtig?
- Konzept der Technischen Schuld (engl. "Technical Debt")
- Broken Windows Theorem
- Code Smells (inkl. Übung)
- Was ist sauberer Code?

Tests als Grundlage von Clean Code

- Unit Tests
- Motivation, Zeichen guter und schlechter Tests, Unit Test Frameworks (inkl. Übung)
- Isolation Frameworks (optional)

- Motivation, Isolation Frameworks (inkl. Übung), Limitationen und Empfehlungen

Regeln, Prinzipien und Best Practices von Clean Code

- Don't repeat yourself (inkl. Übung)
- Keep it short and simple (inkl. Übung)
- Information hiding (Law of Demeter, Tell don't ask)
- Programming to an Interface
- Exkurs: Isolation von Abhängigkeiten
- Modularisierung
- Vorsicht vor Optimierungen
- Principle of Least Surprise
- Favor Composition over Inheritance (inkl. Übung)
- YAGNI - You ain't gonna need it
- Boy Scout Rule
- Root Cause Analysis

SOLID Prinzipien

- Single Responsiblity Principle (inkl. Übung)
- Single Level of Abstraction
- Open Closed Principle (inkl. Übung)
- Liskov Substitution Principle (inkl. Übung)
- Interface Segregation Principle (inkl. Übung)
- Dependency Inversion Principle (inkl. Übung)
- Inversion-of-Control (IOC) Frameworks

Design Patterns (optional)

- Was sind Design Patterns?
- Vor- und Nachteile von Design Patterns
- Ausgewählte Creational, Behavioral und Structural Design Patterns
- Übung zu State Pattern

Refactoring

- Was ist Refactoring?
- Randbedingungen für erfolgreiche Refactorings
- Refactoring-Katalog nach Martin Fowler (inkl. Übungen)
- Optionale, größere Übungen zum Refactoring
- Empfehlungen

C#-spezifische Features für sauberen Code

- String Interpolation
- Auto Properties
- Expression-bodied Members
- Null Checks
- LINQ (inkl. optionaler Übung)
- Extension Methods (inkl. optionaler Übung)
- Record Types

MicroConsult PLUS

- Ihre Übungen und die von MicroConsult entwickelten Lösungen aus dem Workshop stellen wir für Sie zum Download bereit.