Clean Code – wartbare Software entwickeln

Code so entwickeln, dass er auch morgen noch verständlich und änderbar bleibt

01 INTRO

Funktionierender Code ist nur der Anfang. Software wird über Jahre gelesen, erweitert, korrigiert und von unterschiedlichen Entwicklern bearbeitet. Verständlichkeit, klare Verantwortlichkeiten und beherrschbare Abhängigkeiten bestimmen deshalb wesentlich die langfristigen Kosten eines Systems.

Der Kurs vermittelt Clean Code als praktische Engineering-Disziplin statt als Sammlung dogmatischer Regeln. Die Teilnehmer analysieren bestehenden Code, erkennen typische Wartbarkeitsprobleme und verbessern ihn schrittweise, ohne dabei Lesbarkeit mit möglichst kurzer oder vermeintlich eleganter Implementierung zu verwechseln.

DAUER

3 Tage

FORMAT

Präsenz · Live Online · Inhouse

VORKENNTNISSE

Sichere Grundlagen in mindestens einer objektorientierten Programmiersprache

02 TRAININGSPROGRAMM

Inhalte

Was bedeutet Clean Code?

- Verständlichkeit
- Änderbarkeit
- Wartbarkeit
- Testbarkeit
- Einfachheit
- Kontextabhängigkeit
- Regeln versus Prinzipien

Namen als Teil des Designs

- Klassen
- Methoden
- Variablen
- fachliche Begriffe
- Abkürzungen
- Konsistenz
- Intention Revealing Names

Funktionen und Methoden

- Verantwortlichkeiten
- Abstraktionsebenen
- Parameter
- Seiteneffekte
- Rückgabewerte
- Länge
- Command-Query Separation

Klassen und Verantwortlichkeiten

- Kohäsion
- Kopplung
- Single Responsibility
- öffentliche Schnittstellen
- Zustände
- Abhängigkeiten
- geeignete Abstraktionen

Kontrollfluss vereinfachen

- verschachtelte Bedingungen
- Guard Clauses
- komplexe Boolesche Ausdrücke
- Schleifen
- Polymorphie
- frühe Rückgaben
- Lesefluss

Daten und Zustände

- Mutable State
- Immutability
- Datenkapselung
- Data Objects
- Value Objects
- globale Zustände
- temporale Kopplung

Fehlerbehandlung

- Exceptions
- Fehlercodes
- aussagekräftige Fehlermeldungen
- Ressourcen
- Fail Fast
- Fehler nicht verschlucken
- fachliche und technische Fehler

Kommentare und Dokumentation

- guter Code versus Kommentar
- sinnvolle Kommentare
- redundante Kommentare
- veraltete Kommentare
- API-Dokumentation
- Entscheidungen dokumentieren
- Code als Kommunikationsmittel

Code Smells erkennen

- Long Method
- Large Class
- Duplication
- Feature Envy
- Primitive Obsession
- Shotgun Surgery
- Divergent Change
- typische Warnsignale

Abhängigkeiten beherrschen

- Dependency Inversion
- Dependency Injection
- Schnittstellen
- Kopplung
- versteckte Abhängigkeiten
- Testbarkeit
- Overengineering vermeiden

Clean Code im Team

- gemeinsame Konventionen
- Formatter und Linter
- statische Analyse
- Code Reviews
- Definition of Done
- pragmatische Regeln
- technische Diskussionen

Bestehenden Code verbessern

- problematischen Code analysieren
- Verbesserungen priorisieren
- kleine Änderungen
- Tests als Sicherheitsnetz
- Lesbarkeit vergleichen
- Trade-offs diskutieren
- Verbesserungen begründen

03 DURCHFÜHRUNG

Rahmen und Organisation

Dauer	3 Tage
Durchführung	Präsenz oder Live Online, vorzugsweise als Inhouse-Schulung
Schwerpunkt	Inhouse-Schulungen für Unternehmen und Teams
Sprache	Deutsch, englische Fachbegriffe und Dokumentation
Unterlagen	Kursunterlagen, Beispiele und Übungsaufgaben
Technik	Eigener Rechner mit Entwicklungsumgebung für die praktischen Übungen

> **Inhouse-Schulungen**
> Code aus dem eigenen System kann als Grundlage für die Analyse- und Verbesserungsübungen dienen.

04 ZIELGRUPPE

Für wen ist der Kurs gedacht?

Softwareentwickler, technische Tester und Quality Engineers, die wartbaren und verständlichen Code entwickeln oder bestehenden Code verbessern möchten.

> **Voraussetzungen**
> Sichere Grundlagen in einer objektorientierten Programmiersprache.

05 LERNZIELE

Was Sie nach dem Kurs können

Nach dem Kurs können die Teilnehmer:

- Wartbarkeitsprobleme in Code erkennen
- Verantwortlichkeiten klarer strukturieren
- Namen, Methoden und Klassen verständlicher gestalten
- unnötige Kopplungen und komplexe Kontrollflüsse reduzieren
- Code Smells systematisch erkennen
- Clean-Code-Prinzipien pragmatisch statt dogmatisch einsetzen

06 FAQ

Häufige Fragen

Gibt es objektiv sauberen Code?

Nicht vollständig. Viele Prinzipien lassen sich begründen, ihre konkrete Anwendung hängt aber von Sprache, System, Team und Kontext ab. Der Kurs behandelt Clean Code deshalb nicht als starres Regelwerk.

Wird im Kurs programmiert?

Ja. Ein wesentlicher Teil besteht aus der Analyse und Verbesserung konkreten Codes.

Worin unterscheidet sich der Kurs vom Refactoring-Kurs?

Dieser Kurs behandelt die Eigenschaften wartbaren Codes. Der Kurs zu Refactoring und Legacy Code konzentriert sich auf den kontrollierten Prozess, bestehenden Code unter Erhaltung seines Verhaltens zu verändern.

UC IT Service · Ulrich Cuber · kontakt@uc-it.de
https://www.uc-it.de/coaching/clean-code/