Publicatiedatum : 17/02/2026

Opleiding : Schone code

Praktijkcursus - 2d - 14u00 - Ref. CCJ
Prijs : 1280 € V.B.

NEW

Deze cursus is gericht op het transformeren van uw benadering van ontwikkeling rond twee fundamentele pijlers: codekwaliteit en beveiliging. Van het produceren van duidelijke, efficiënte code tot het structureren van eenvoudige, consistente functies, u zult zien hoe u code kunt produceren die iedereen kan begrijpen.

Pedagogische doelstellingen

Aan het einde van de training is de deelnemer in staat om:

- Technische schuld beoordelen en verminderen
- Leesbare en expressieve code produceren
- Eenvoudige, samenhangende functies structureren
- De SOLID-principes in de praktijk brengen
- Een effectieve teststrategie implementeren
- Verouderde code volledig veilig refactoren
- De kwaliteit van code industrialiseren
- AI integreren als ontwikkelingsassistent

Doelgroep

Objectprogrammeerders.

Voorafgaande vereisten

Ruime programmeerervaring, idealiter in een objectgeoriënteerde taal. Goede beheersing van een IDE en basiskennis van unit testen.

Praktische modaliteiten

Praktisch werk

Analyse, discussie en praktisch werk.

Opleidingsprogramma

DEELNEMERS

Objectprogrammeerders.

VOORAFGAANDE VEREISTEN

Ruime programmeerervaring, idealiter in een objectgeoriënteerde taal. Goede beheersing van een IDE en basiskennis van unit testen.

VAARDIGHEDEN VAN DE CURSUSLEIDER

De deskundigen die de cursus leiden zijn specialisten op het betreffende vakgebied. Zij werden geselecteerd door onze pedagogische teams zowel om hun vakkennis als hun pedagogische vaardigheden voor elke cursus die zij geven. Zij hebben minstens vijf tot tien jaar ervaring in hun vakgebied en oefenen of oefenden verantwoordelijke bedrijfsfuncties uit.

BEOORDELINGSMODALITEITEN

De cursusleider beoordeelt de pedagogische vooruitgang van de deelnemer gedurende de gehele cursus aan de hand van meerkeuzevragen, praktijksituaties, praktische opdrachten, ...
De deelnemer legt ook van tevoren en naderhand een test af ter bevestiging van de verworven kennis.

1 Van programmeur tot softwarevakman

- Technische schuld: de werkelijke kosten van foutieve code meten.
- De padvindersregel: laat de code schoner achter dan je hem gevonden hebt.
- De impact van AI: hoe ChatGPT en Copilot het spel veranderen.

Praktisch werk

Collectieve analyse van een "vuile" codefragment. Identificatie van codegeuren en AI-ondersteund herschrijven om resultaten te vergelijken.

2 De kunst van naamgeving en duidelijkheid

- De bedoeling is vooral: namen kiezen die het "waarom" onthullen en niet het "hoe".
- Elimineer dubbelzinnigheid: vermijd mentale koppelingen en onuitspreekbare namen.
- Moderne conventies: het benoemen van klassen en methoden in het tijdperk van moderne frameworks.

Praktisch werk

Refactoring van cryptische variabelen en functies (bijv. d, list1, proc()) naar duidelijke bedrijfsconcepten.

3 Narratieve functies en complexiteit

- Single Responsibility Principle (functies): een functie mag maar één ding doen.
- Abstractie-niveaus: haal het hoge niveau (business) en het lage niveau (technische details) niet door elkaar.
- Argumenten en neveneffecten: waarom booleaanse vlaggen vermijden en argumenten beperken.
- Command Query Separation (CQS): scheidt commando's van query's.

Begeleid praktisch werk

Neem een 100-regelige "God Functie" en verdeel deze in 10 atomaire, leesbare functies.

4 Toegepaste SOLID-architectuur

- SRP (Responsabilité Unique): samenhang van klassen en modules.
- OCP (open/closed): uitbreiden zonder te wijzigen (het gebruik van polymorfisme).
- LSP & ISP: Liskov-substitutie en scheiding van interfaces om [[Fat Interfaces]] te vermijden.
- DIP (Dependency inversion): sterke ontkoppeling om testen te vergemakkelijken.
- DRY (Don't Repeat Yourself): nuance tussen duplicatie van code en duplicatie van kennis.

Praktisch werk

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

PEDAGOGISCHE EN TECHNISCHE MIDDELEN

• De gebruikte pedagogische middelen en cursusmethoden zijn voornamelijk: audiovisuele hulpmiddelen, documentatie en cursusmateriaal, praktische oefeningen en correcties van de oefeningen voor praktijkstages, casestudies of reële voorbeelden voor de seminars.
• Na afloop van de stages of seminars verstrekt ORSYS de deelnemers een evaluatievragenlijst over de cursus die vervolgens door onze pedagogische teams wordt geanalyseerd.
• Na afloop van de cursus wordt een presentielijst per halve dag verstrekt, evenals een verklaring van de afronding van de cursus indien de stagiair alle sessies heeft bijgewoond.

TOEGANGSMODALITEITEN EN TERMIJNEN

De inschrijving dient 24 uur voor aanvang van de cursus plaatsgevonden te hebben.

TOEGANKELIJKHEID VOOR MINDERVALIDEN

Is voor u speciale toegankelijkheid vereist? Neem contact op met mevr. FOSSE, contactpersoon voor mindervaliden, via het adres psh-accueil@ORSYS.fr om uw verzoek en de haalbaarheid daarvan zo goed mogelijk te bestuderen.

5 Testgestuurde ontwikkeling (TDD)

- De Red-Green-Refactor-cyclus: de adem van de ontwikkelaar.
- Testkwaliteit (FIRST): Snel, Onafhankelijk, Herhaalbaar, Zelf-Validerend, Tijdig.
- Outillage moderne : focus sur les standards actuels (JUnit 5, Jest, Pytest).
- Mocking versus stubbing: effectief gebruik van Mockito of Jest Mocks.

Praktisch werk
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 beheren (nieuwe module)

- Definitie van legacy: code zonder tests.
- Technieken om afhankelijkheden te doorbreken: hoe het ontestbare te testen.
- Le refactoring sécurisé : utiliser l'IDE pour refactorer sans casser. Technique du "Sprout Method/Class".

Praktisch werk
"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 Schone code op schaal (tools & AI)

- Automatische code voering (ESLint, Checkstyle).
- SonarQube/Checkmarx/SonarLint: detecteer technische schuld en beveiligingsfouten in realtime.
- Code Coverage: de mythes en realiteit van code coverage.
- Gebruik AI om relevante documentatie te genereren (JSDoc/JavaDoc).
- Prompter efficiënt met AI om unit tests te genereren.

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

Data en plaats

KLAS OP AFSTAND
2026 : 15 juni, 17 sep., 7 dec.

PARIS LA DÉFENSE
2026 : 8 juni, 10 sep., 30 nov.