Clean Code

Durée 3 jour(s) (CLEAN-CODE-03)

Concevoir et écrire un code propre, améliorer un code existant

Description

L'introduction des méthodes agiles a permis de mieux construire des logiciels conformes aux besoins réels, et d'améliorer le processus de delivery. Le clean code est un ensemble complémentaire de pratiques techniques, permettant d'assurer la pérennité du logiciel construit. Un code propre est en effet la condition sine qua non d'un logiciel robuste (coût de maintenance limité) et évolutif (pouvant être adapté aux nouveaux besoins).

Formation animée en présentiel

La formation en présentiel se déroule sur des jours consécutifs

Formation disponible en mode "formation à distance"

La formation à distance se déroule de préférence sur des jours consécutifs (contactez nous si besoin de décomposer en demies journées)

En inter-entreprises, l'outil de visio-conférence privilégié est Microsoft Teams

En intra-entreprises, on privilégie Zoom mais Microsoft Teams est également proposé

Objectifs

- Approfondir les principes de programmation orientée objet pour écrire du code conforme à l'état de l'art
- Concrétiser et mettre en pratique ces principes avec des exemples simples et frappants
- Mettre sous contrôle la dette technique par le refactoring
- Apprendre les techniques spécifiques aux applications legacy
- Acquérir une vision synthétique des méthodes de conception propre les plus utilisées

Public

- Développeurs
- Architectes

Profils ayant de l'expérience en Java.

Prérequis

Aucun

Répartition

50% Théorie, 50% Pratique

Evaluations des acquis

L'évaluation des acquis de la formation se fera en séance au travers d'ateliers, d'exercices et/ou de travaux pratiques. Dans le cas d'une formation officielle éditeur, veuillez nous consulter afin que nous vous fassions part des modalités d'évaluation.

A l'issue de la formation, vous sera transmis une évaluation à chaud de l'action de formation qui vous permettra de nous faire part de vos retours quant à votre expérience apprenant avec Zenika.

Ressources pédagogiques

Les ressources pédagogiques proviennent de productions des équipes Zenika et/ou de la documentation éditeur dans le cas d'une formation "Officielle". Les documents sont en français ou en anglais.

RQTH et ma formation Zenika

Si vous êtes sujet à un handicap, prenez contact avec nos équipes pour que nous puissions définir ensemble comment nous pourrons aménager la session afin que vous puissiez vivre une expérience en formation inchangée.

Programme

L'impératif de qualité logicielle

- Conséquences d'une qualité insuffisante
- Le cycle infernal: écriture, réécriture
- Le concept de dette technique
- Outils et processus: nécessaires, mais pas suffisants
  - Outils de contrôle de qualité
  - Processus et formattage
  - Limitations
- Qu'est-ce que du code propre ?

Principes de conception propre

- Principes généraux
  - Principes fondateurs de la POO
  - Les quatre principes de Kent Beck
  - Importance du nommage
  - Le bon sens par les acronymes: YAGNI/KISS/DRY/POLA
  - Quelques principes de programmation fonctionnelle
- Minimiser le couplage, maximiser la cohésion
  - Les principes SOLID
  - Cohésion et couplage
  - Stabilité et instabilité
- Supple design
  - Intention-Revealing Interfaces
  - Side-Effect-Free Functions
  - Defensive Programming
  - Conceptual Contours
  - Standalone Classes et Closure of Operations
  - Declarative Style of Design
  - Bonus: le principe de symétrie

Améliorer la qualité du code existant: smells et refactors

- Les concepts de smell et de refactor
- Les smells de Martin Fowler
  - Duplicated method
  - Duplicated class
  - Long method
  - Long class
  - Primitive obsession
  - Bref aperçu d'autres smells
- Uneffective Java
  - Egalité.. ou pas
  - Immutabilité à trous
  - Obsolescence instantanée
  - Cachez cette exception que je ne saurais voir
  - ArrayList obsession
- Design faible
  - Modules techniques: Service-Dao-Entity
  - Generate getters and setters
  - La mort par imbrication: l'escalier du diable
  - Paquets dépaquetés
  - Etre ou ne pas être: la relation inappropriée Is-not-a
  - Javadoc et tests alibi
  - L'éditorialiste: l'intelligence enfouie dans les commentaires

    - La nuit des codes vivants
- Modifier proprement une application legacy

Panorama des autres méthodes de conception propre

- La conception Test-first au service de la qualité
- Software Craftsmanship
- Briques de conception standard
- Le domaine métier comme noyau du logiciel: l'approche Domain-driven Design