Restructuration et amélioration incrémentale du code (Refactoring)

Durée: 2 jours
Public cible: Professionnel TI

À qui s'adresse cette formation ?

Développeurs

Objectifs de la formation

Mettre en œuvre les meilleures techniques de conception de classes de l'industrie afin d'améliorer la structure, la lisibilité, la maintenance et l'évolutivité d'un logiciel objet.

Description sommaire

Améliorez la conception et la réutilisation de vos classes (nouvelles et existantes) dans vos projets objet

Contenu du plan de cours

- Techniques d'amélioration de la conception de codes existants sans altération du comportement externe, principes et critères d'application des techniques de refactoring
- Présentation d'un catalogue de patrons (Patterns) d'implémentation pour les classes
- Réorganisation des méthodes : comment les extraire, les remplacer ou les ajouter
- Réorganisation des attributs : comment créer, changer ou remplacer des accesseurs, des valeurs, des références, des données observées, des associations et des énumérations
- Déplacement des responsabilités à l'aide des classes en ligne, déléguées ou intermédiaires
- Simplification des expressions conditionnelles : utilisation de sous-classes, patrons état / stratégie, objet nul et assertion
- Simplification des appels aux méthodes pour construire des interfaces et des fabrications (Factories)
- Déplacement de méthodes dans une hiérarchie d'héritage : comment extraire et déplacer des attributs et des méthodes

Approche et méthode pédagogique

- Exposés
- Démonstrations
- Exercices dirigés et individuels

La répartition du contenu est approximativement : matériel 35% et laboratoires 65%

Prérequis

- Expérience préalable avec un langage de programmation orienté objet.
- Connaissances de base en développement logiciel et en conception de classes.
- Expérience pratique en développement (souhaitable pour tirer pleinement profit des exercices).

Recommandations

Réviser : les principes de base du code propre les concepts de POO

Identifier vos enjeux : code difficile à maintenir duplication complexité excessive

Apporter (si possible) : un exemple de code réel à améliorer