DAWAN

Plan de la formation

www.dawan.fr

# Formation C# Intermédiaire : Optimiser le modèle objet et utiliser les design patterns

| | |
|---|---|
| **Durée :** | 5 jours (35 heures) |
| **Tarif :** | 2 395,00 € (standard)<br>1 916,00 € (remisé) |
| **Public :** | Développeurs C# |
| **Pré-requis :** | Avoir suivi la formation C# Initiation+Approfondissement ou notions équivalentes |
| **Objectifs :** | Maîtriser les fondements de la POO - Comprendre la décomposition d'une application d'entreprise en objets (conception/design OO) - Appliquer les principes de regroupement, de structuration et de communication entre les objets d'un système complexe - Concevoir des systèmes OO de manière à  favoriser la maintenabilité et faciliter le changement dans un contexte itératif - Appliquer les principes S.O.L.I.D. - Comprendre certains modèles de conception d'entreprise (Repository, Factory, DTO) - Connaître la place et les différences entre les styles architecturaux - Connaître quelques modèles architecturaux (DDD, Clean Architecture ...) - Concevoir des applications faiblement couplées et cohésives - Apprendre à  implémenter des designs patterns |
| **Sanction :** | Attestation de fin de formation mentionnant le résultat des acquis |
| **Référence:** | .NE101050-F |
| **Note de satisfaction des participants:** | 4,88 / 5 |

**Maîtriser les fondements de la conception objet**

Encapsulation : intérêt, bonnes pratiques
Agrégation d'objets
Héritage : cas d'usage, préférence pour la composition
Polymorphisme : ad-hoc, sous-typage, types paramétriques
Objets Valeurs (Value Objects)
Cercle vertueux de l'ignorance

**Atelier : construire un schéma de classes cohérent**

**Gérer l'interaction entre les objets du système**

Tell don't ask
Gestion des dépendances
Découpage des règles d’affaires basé sur l’interaction
Conception basée sur les comportements
Loi de Déméter

**Atelier : implémentation de patterns de comportements**

**Concevoir un domaine et découper des objets**

Conception par concepts plutôt que par données : concepts, types d'objets
Architecture Héxagonale
Présentation des principes SOLID
Principe de la responsabilité unique (SRP)
Principe de l’ouverture-fermeture (OCP)

**Atelier : multiples exemples de mauvaise/bonne implémentation**

**Introduire une abstraction**

Métrique de l'Abstraction-Instabilité (R. C. Martin)
Principe de substitution de Liskov (LSP)
Composition versus héritage
Principe de la ségrégation des interfaces (ISP)

**Atelier : analyse d'un code et présentation des métriques - ré-écriture d'exemples concrets**

**Concevoir une application en couches**

Conception modulaire
Conception d'un domaine d'affaires (aperçu du DDD)
Séparation de l’infrastructure (persistance, UI, ORM, etc.)
Principe d’inversion des dépendances (DIP)
Entrepôts référentiels (Repositories)
Objet de transport (DTO)
Présentation de la clean architecture

**Atelier : implémentation d'une application en couches**

**Comprendre et appliquer les design patterns**

Historique et ouvrages de référence
Domaines d'application
Comment appliquer les Design Patterns

**Générer des instances**

Factory et Abstract Factory pour la création sous condition
Singleton et dérivé : maîtrise des ressources disponibles

**Organiser les structures de données**

Le Composite, comment simplifier les listes
Proxy et Adapter, les interfaces de l'accès aux méthodes
La Facade : clarifier un composant

**Maîtriser le comportement des objets**

Strategy : l'usine à méthodes
Observer : l'événementiel sans événements
Chaîne de responsabilité et arbres de responsabilité
Visiteur et accès : maîtrise de la collaboration
Aperçu d'autres Design Patterns

**Atelier : implémentation des différents design patterns étudiés**