

## Formation Java SE Intermédiaire : Conception objet avancée

<b>Durée :</b>	3 jours
<b>Public :</b>	Développeurs Java
<b>Pré-requis :</b>	Avoir suivi la formation Java Initiation+Approfondissement ou notions équivalentes 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.
<b>Objectifs :</b>	
<b>Sanction :</b>	Attestation de fin de stage mentionnant le résultat des acquis
<b>Taux de retour à l'emploi:</b>	Aucune donnée disponible
<b>Référence:</b>	JAV100952-F
<b>Note de satisfaction des participants:</b>	Pas de données disponibles

### 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 applicaion en couche**