Programmation Orientée Objet avec JAVA

3 jours

Objectifs

Ce stage permet aux participants d'aborder l'intégralité des principes fondamentaux du développement objet. Il est particulièrement recommandé aux auditeurs placés dans des situations d'encadrement de sous-traitants, et souhaitant disposer d'éléments techniques solides leur permettant de dialoguer plus efficacement avec leurs partenaires. Ce stage, très pratique, permet également à des développeurs d'appréhender la programmation orientée objet avec le langage Java.

Contenu de la formation

- Présentation générale
- Aspects syntaxiques, types et expressions
- Méthodes et instructions
- Utilisation de l'abstraction
- Utilisation de l'héritage
- Utilisation du mécanisme d'interface
- Développement de classes
- Développement d'interfaces
- Développement de classes dérivées
- Les exceptions

Participants

Consultants fonctionnels, chefs de projets, souhaitant acquérir une vision précise des techniques employées par leurs collaborateurs ou sous-traitants. Développeurs traditionnels voulant évoluer " en douceur " vers l'objet.

Pré-requis

Aucune connaissance particulière.

Travaux pratiques

Un cas d'étude à dominante professionnelle est utilisé comme fil directeur durant ce stage. Chaque apport théorique fait l'objet d'une mise en application immédiate, afin de permettre une bonne compréhension par les auditeurs.

Programmation Orientée Objet avec JAVA

Programme

» Présentation générale

Principes fondateurs de l'objet : encapsulation. héritage, polymorphisme.
Présentation générale du langage: machine virtuelle, interpréteur, compilateur, documentation
Distributions de Java.

» Aspects syntaxiques, types et expressions

Structuration syntaxique d'une application Java.
Exemple de syntaxe sur une application simplifiée.
Notion de type. Comparaison des types primitifs et des types complexes.
Utilisation simple des types primitifs : nombres entiers, flottants, caractères et booléens.
Notion d'expression.
Déclarations : variables et constantes.
Utilisation des opérateurs avec les objets.
Utilisation des tableaux.
Conversion d'une variable de type primitif en objet.

» Méthodes et instructions

Syntaxe d'appel des méthodes.
Définition et utilisation des méthodes.
La surcharge des méthodes.
Instructions de contrôle principales : if, while, for, return, break.

» Utilisation de l'abstraction

Exemple simple d'utilisation d'un objet : déclaration et instanciation.
Utilisation des constructeurs d'objets.
Utilisation de l'interface programmatique des objets : exemple des classes Date et Calendar.
Une classe très utilisée : la classe String.
Utilisation de la classe StringBuffer : exemple d'utilisation de la surcharge de méthodes.

» Utilisation du mécanisme d'héritage

Rappel du principe d'héritage.
Utilisation de l'héritage.
Exemple de graphe d'héritage.
La classe Object: classe parente de toutes les classes.
Utilisation du polymorphisme.
Comportement des méthodes et typage.
Généricité: principe et mise en application à partir de la classe Vector.

» Utilisation du mécanisme d'interface

L'interface: un contrat entre un fournisseur de service et un consommateur.
L'interface utilisée comme mécanisme de contrainte.
Syntaxe associée aux interfaces.
Exemple d'implémentation multiple d'interfaces.
Exemples avancés d'utilisation d'interfaces.

» Développement de classes

Approche méthodologique, analyse statique, dynamique, métier.
Notation UML : diagramme de classe, de séquence, de cas d'utilisation.
Squelette d'une classe : constituants de base, outils de génération automatique.
La visibilité: mécanisme de contrainte d'accès aux propriétés et méthodes.
Organisation en packages.
Ecriture des constructeurs.
Constructeur par défaut.
L'auto-référence this.
Champs et méthodes statiques.
La méthode main.

» Développement de classes dérivées

Rappels des principes.
Méthodes et classes abstraites.
Différence entre classes abstraites et interfaces.
Droit d'accès aux champs et héritage.
Enchaînement des constructeurs et héritage.
Redéfinition et surcharge.

» Les exceptions

Principes généraux.
Détection, constat et notification d'une situation d'erreur.
Report d'une exception : clause throws de la signature, bloc try/catch.
Déclenchement d'exception en cas d'erreur fonctionnelle.
La délégation d'exception
Exemples de gestion d'exception, de délégation.