Programmer par objets avec Java

poj

Spoc pratique
Travaux pratiques continus

Durée
3 jours

Votre animateur
Philippe Brutus ou Olivier Charles

Code Kertify : **POJ**

S'inscrire

www.kertify.com/inscription

Apprendre à programmer par objets avec Java

La programmation par objets repose sur une approche modulaire qui définit les éléments du domaine plutôt que les fonctionnalités de l'application. Leur spécification par des interfaces facilite le dialogue entre la MOA et la MOE.

Vous apprendrez les principes fondamentaux du développement par objets en programmant des exemples concrets.

Les objectifs de la formation

- Comprendre les principes de l'approche de la programmation par objets
- Appréhender la syntaxe du langage Java
- Maîtriser les échanges techniques entre équipe de développement et maîtrise d'ouvrage
- Maîtriser la spécification d'applications par objets-métier

Participants

- Développeur
- Chef de projets voulant acquérir la vision des techniques employées par ses équipes de développement ou ses sous-traitants.

Prérequis

- Des connaissances de base en programmation

Le programme de notre formation

1. Introduction à la programmation par objets

- Principes de l'approche par objets : fusion données-traitements, encapsulation, abstraction, héritage.
- Le langage Java, les outils et les bibliothèques de composants prédéfinis.
- Les distributions de Java.

2. Aspects syntaxiques, types et expressions

- Structure d'une application Java
- Syntaxe d'utilisation d'objets
- Syntaxe de définition d'une classe pour l'implantation d'objets
- Notion de type de données, types de base et types objet.
- Utilisation des types de base et des classes correspondantes, conversions de types
- Notion d'expression
- Déclaration de variable et de constante
- Notion de caractéristiques de classe et d'objets (champs et méthodes statiques)

- Utilisation des tableaux
- Conventions d'écriture

3. Méthodes et instructions

- Définition et utilisation des méthodes
- Structures de contrôle : if, if-else, for, while, do-while, switch-case, try-catch, break et return
- Notion de bloc d'instructions
- Polymorphisme
- Surcharge de méthode

4. Abstraction

- Déclaration et instanciation d'objet, délégation
- Utilisation des constructeurs d'objets
- Documentation en ligne
- Interface de programmation des objets (API)
- La classe String et les particularités des chaînes de caractères
- La classe StringBuffer et la surcharge de méthodes

5. Héritage

- Principe d'héritage et terminologie
- Utilisation de l'héritage
- La classe Object et la généricité
- Utilisation du polymorphisme
- Typage des objets
- Comportement des méthodes et typage
- Généricité des classes collections et classes paramétrées (generics)

6. Interface

- Interface implicite et explicite d'une classe
- Syntaxe associée aux interfaces explicites
- Utilisation des références d'interfaces : flexibilité, limitation de portée, polymorphsme
- Exemple d'implantations multiples d'interfaces
- Synthèse sur l'intérêt des interfaces pour les méthodes
- Utilisation des interfaces pour les constantes
- Exemples avancés d'utilisation d'interfaces

7. Développement de classes

- Approche méthodologique, analyse statique, dynamique, métier.
- Notation UML : diagramme de classes, d'état/transition, de séquence.
- Squelette d'une classe : constituants de base, outils de génération automatique.
- Compléments sur les droits d'accès.
- Organisation en paquets (package) et contraintes liées aux paquets.
- Ecriture des constructeurs.
- Constructeur par défaut.
- Compléments sur l'écriture des constructeurs.
- L'auto-référence "this".

- Champs et méthodes statiques.
- La méthode "main".

8. Développement d'interfaces

- Rappels et compléments sur les principes.
- Syntaxe associée aux interfaces, cas des constantes.
- Définition d'interfaces pour les méthodes.
- Implantation et extensions multiples d'interfaces.
- Implantation partielle d'interface.
- Exemples sur l'utilisation d'interfaces.

9. Développement de classes dérivées

- Rappels des principes.
- Approche méthodologique pour le découpage en classes.
- Méthodes et classes abstraites.
- Classes abstraites et interfaces.
- Droit d'accès aux champs et héritage.
- Enchaînement des constructeurs et héritage.
- Redéfinition et surcharge.

10. Les exceptions

- Principes et cinématique générale.
- Détection, constat et notification d'une situation exceptionnelle.
- Report d'une exception : clause "throws" de la signature, bloc try-catch.
- Exceptions non vérifiées.