réf. di31

CONCEPTION ET PROGRAMMATION ORIENTÉES OBJET

INFORMATIONS

CONTACT
03 88 47 10 96
mfo@metaformose.org

A QUI S'ADRESSE LA FORMATION ?
- Développeurs issus du monde de la programmation procédurale
- Développeurs, analystes programmeurs et chefs de projets «anciennes technologies»

MÉTHODES ET OUTILS PÉDAGOGIQUES
- Formation en présentiel, interactive axée sur la pratique pour une meilleure compréhension et application
- Supports vidéo et exercices
- Documents informatiques et papiers

PREREQUIS
- Disposer d'une première expérience de la programmation en environnement professionnel

NOMBRE DE PARTICIPANTS
2 à 8 personnes

DURÉE DE L'INTERVENTION
4 journées soit 28 heures
9h-12h30 et 13h30h-17h

EVALUATION
- Contrôle des connaissances en cours de formation, tests, questionnaires
- Fiche d'évaluation et de satisfaction stagiaire
- Attestation individuelle de formation

INTERVENANTS
- Formateurs seniors experts en développement

LES OBJECTIFS DE LA FORMATION

» Comprendre la notion d'objet et les concepts associés
» Disposer des connaissances générales nécessaires à l'apprentissage du développement Objet
» Comprendre la notion d'Objet et les concepts associés
» Être capable de modéliser une application à l'aide d'UML 2
» Identifier les apports des Design Patterns en phase de conception

LE PROGRAMME DE LA FORMATION

1. Qu'est-ce que la conception d'un programme informatique
- Enjeux et défis : des systèmes de plus en plus complexes
- Les acteurs d'un projet informatique
- Importance du choix de la méthode et des outils
- L'apport d'UML dans la modélisation de programmes informatiques

2. Structurer un programme
- Les bonnes pratiques de l'écriture de code
- Notions sur le «Clean Code»
- Commentaires utiles
- L'approche structurée
- Modularité du code par ajout de bibliothèques
- Couplage faible Vs. Cohérente forte
- Gestion des bibliothèques
- Gestion des données du programme

3. Programmation structurée et programmation orientée objet
- Pourquoi travailler avec des objets
- Dualité données et traitement dans l'approche orientée objet
- Concepts de classe, héritage, polymorphisme
- Les avantages de l'encapsulation

4. L'approche objet
- Les objectifs de la programmation Objet
- L'instanciation ou la création d'un objet à partir d'une classe
- Utilisation de constructeurs
- Libération des ressources à l'aide des destructeurs
- Les concepts objet : les objectifs du monde Objet, les classes et les objets, les attributs, les méthodes, l'encapsulation, l'instanciation
- Traduction des concepts Objet en langage : les packages et les espaces de noms, les classes, les méthodes et leur visibilité, les attributs et leur visibilité, l'instanciation, l'appel de méthodes et la référence aux variables
- Organisation par package et espace de noms

5. Héritage et encapsulation
- Comment spécialiser une classe et réutiliser du code
- Un exemple concret pour comprendre l'utilité de l'héritage
- Redéfinir une méthode dans une classe fille avec le polymorphisme
- Notion de classes et de méthodes abstraites

6. Introduction à UML
- UML un standard bien établi dans le monde industriel
- L'importance de la modélisation dans les projets complexes
- Présentation des différents diagrammes et points de vues
- Présentation des outils de modélisation : Enterprise Architect , Magic Draw, Visual Paradigm

7. Concevoir le système logiciel à l'aide d'UML
- Définir la plate-forme technique : définir l'architecture matérielle (diagramme de déploiement), choisir le framework logiciel
- Concevoir un code source répondant aux exigences, maintenable et évolutif
- Définir une architecture du code: le pattern en couches MVC, étendu au système entier
- Concevoir les attributs: attributs identifiants et dérivés- association entre classes (diagramme de classe)
- Concevoir les traitements et la communication entre classes (diagramme de séquence) : utiliser les scénarios de cas d'utilisation – répondre aux exigences fonctionnelles, séparer les préoccupations selon MVC
- Affiner la structuration du code source : affiner la structuration en packages, factoriser du code avec la généralisation – du bon usage de l'héritage, faire communiquer les classes en limitant les dépendances : utilisation des interfaces et des singletons – pattern de communication requête/notification, gérer les états
- Concevoir les composants déployables : définir les composants et leurs interfaces (diagramme de composant), définir le déploiement des composants