Conception orientée objet ingénierie logicielle objet

-Réference: **SII-311** -Durée: **4 Jours (28 Heures)**

Les objectifs de la formation

- Pourquoi utiliser des technologies à objets ?
- Les défis de la nouvelle informatique : modularité (Plug-Ins), réutilisabilité, évolutivité
-
- L'utilisation de bibliothèques de composants
- Comment l'approche Objet répond à ces défis ?
- Dans quel état d'esprit aborder un problème Objet ?
- Les acquis provenant des autres domaines de l'informatique et des autres disciplines
-

A qui s'adesse cette formation ?

POUR QUI :

- Développeurs, chefs de projets souhaitant se former à la conception orientée Objet.

PRÉREQUIS :

- Connaissances de base en conception d'applications et en développement logiciel.
- .

Programme

- **Qu'attendre de l'approche Objet ?**
  - Pourquoi utiliser des technologies à objets ? Les défis de la nouvelle informatique : modularité (Plug-Ins), réutilisabilité, évolutivité.
  - L'utilisation de bibliothèques de composants.
  - Comment l'approche objet répond à ces défis ? Dans quel état d'esprit aborder un problème objet ? Les acquis provenant des autres domaines de l'informatique et des autres disciplines.
- **Les concepts de base de l'approche Objet**
  - Les objets : une dualité procédure/donnée.
  - Les classes comme modèles de structure et de comportement des objets, les instances comme représentants des classes.
  - Les méthodes, des procédures définies dans les classes et utilisées par les instances.

Programme

- Les interactions entre objets par envois de messages.
- Comment les messages sont interprétés par les objets ? L'héritage.
- Héritage et typage des variables dans les langages fortement typés (C++, Java).

- **Diagrammes et représentation des objets à l'aide d'UML**
  - Les principaux diagrammes (diagrammes de classe, diagrammes de séquence) et leur utilisation pour la conception Objet.
  - Les outils de notation et représentation des objets : prise en main d'un modeleur du marché.

- **Les grands principes de la conception objets**
  - Que met-on sous la forme d'un objet ? Principe de réification.
  - Critères à appliquer pour décider de ce qui doit être mis sous forme Objet.
  - Les erreurs à éviter.
  - Comment structurer un logiciel objet ? Principe de modularité et de décomposition des domaines.
  - Comment structurer un ensemble de classes ? Principe d'abstraction et de classification.
  - Comment penser l'interaction entre objets ? Principe d'encapsulation et d'autonomie.
  - Analyser des systèmes complexes en termes de communications.
  - La démarche générale.
  - Les erreurs à éviter.
  - Critères à appliquer pour disposer de "bonnes" hiérarchies de classes.
  - Les erreurs à éviter.

- **Comment aborder un logiciel objet ?**
  - Les principes de développement.
  - Du développement en spirale au développement incrémental.
  - Identification des entités du domaine et description des interactions.
  - Réutilisation et évolutivité des programmes.
  - Concevoir par objets, ce n'est pas utiliser un outil Objet ! Les erreurs à éviter.

- **De la conception à l'implémentation**
  - Comment traduire les diagrammes de classe UML dans des langages de programmation et dans des bases de données ? Les principes de mise en oeuvre d'applications objet.
  - L'importance du distribué.
  - Modèles clients-serveurs généralisés.
  - Les grandes plateformes objets actuels : les technologies .

Programme

- NET de Microsoft et JEE de SUN.
- Comparaison de leurs points forts et de leurs points faibles.
- L'importance du distribué.
- Bibliothèques de classes.
- Langages de programmation et d'utilisation de composants.

- **L'approche par frameworks et composants**
  - Le problème du cycle de vie des logiciels.
  - Les problèmes d'évolution et de maintenance nécessitent une approche logicielle permettant l'évolution.
  - L'approche par frameworks et composants, qui est fondée sur la pensée Objet, est une réponse à cette nécessité.
  - Comment concevoir et réaliser des applications rapidement à partir de frameworks et de composants réutilisables ? Comment intégrer des composants logiciels dans un framework existant ? Comment construire des frameworks ? Savoir reprendre une application existante pour la transformer en framework et la rendre ainsi évolutive.
  - Grandes classes de frameworks.
  - Les modèles de composants actuels.

- **Les Design Patterns**
  - Comment réutiliser de l'expérience lors de la conception et du développement d'applications objets ? Les Design Patterns ou "patrons de conception" comme solutions logicielles issues de problèmes généraux récurrents.
  - Les différents types de Design Patterns.
  - Exemple de Design Patterns.
  - Avantages et limites des Design Patterns.
  - Comment utiliser pratiquement des Design Patterns ? Apprendre à mettre en oeuvre des Design Patterns par la pratique.

(+212) 5 22 27 99 01

(+212) 6 60 10 42 56

Contact@skills-group.com

Nous sommes à votre disposition :
De Lun - Ven 09h00-18h00 et Sam 09H00 – 13H00

Angle bd Abdelmoumen et rue Soumaya, Résidence Shehrazade 3, 7éme étage N° 30
Casablanca 20340, Maroc