CONCEPTION ET PROGRAMMATION OBJET

REF **IM18**

- ***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***

4

Prix : 2?195 € € / HT

OUTILS PÉDAGOGIQUES

PUBLIC VISÉ

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

MODALITÉS D'ÉVALUATION

MODALITÉS DE FINANCEMENT

MODALITÉS ET DÉLAIS D'ACCÈS

OBJECTIFS PÉDAGOGIQUES

ACCESSIBILITÉ

LES POINTS FORTS DE LA FORMATION

PRÉ-REQUIS

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

MODALITÉS ET DÉLAIS D'ACCÈS

ATTESTATION OBTENUE

EFFECTIF DE LA FORMATION

CERTIFICATION

MODALITÉ PÉDAGOGIQUE

Cours dispensé en mode présentiel avec une alternance d'apports théoriques et méthodologiques, et de mises en situations pratiques

PROCHAINES SESSIONS

Le 24/01/2022
Le 02/05/2022
Le 24/01/2022
Le 02/05/2022
Le 24/01/2022
Le 02/05/2022

PROGRAMMES DE CONCEPTION ET PROGRAMMATION OBJET

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

CONCEPTION ET PROGRAMMATION OBJET

REF **IM18**

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

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

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

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

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

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 (diagramme de 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 (diagramme d'états)
- Concevoir les composants déployables : définir les composants et leurs interfaces (diagramme de composant), définir le déploiement des composants (diagramme de déploiement)

INTRODUCTION AUX DESIGN PATTERNS

- Principes des solutions de conception cataloguées

- Méthodologie : définition des besoins techniques, des classes « types » du pattern, des collaborations entre classes
- Présentation des patrons de conception : origine, les 3 familles (création, structuration et comportement), autres patrons
- Présentation des principaux patrons de conception de chaque catégorie
- Documentation d'un patron de conception et présentation des différents diagrammes UML utilisés
- Bonnes pratiques : comment vous aider à choisir le bon patron pour un problème donné