Identification du module

Numéro de module	**226A**
Titre	Programmer selon les principes de la programmation orientée objet (sans héritage)

Compétence	Concevoir un modèle orienté objet à partir d'un cahier des charges qui ne nécessite pas l'utilisation des relations de réalisation ou de spécialisation/généralisation, implémenter et documenter les classes, puis les tester à l'aide d'outils adaptés.

Objectifs opérationnels

1	Être capable de comprendre un modèle orienté objet et de l'étendre avec ses propres classes métiers ou techniques.
2	Connaître la notation UML (*Unified Modeling Langage*) pour la représentation des aspects statiques et dynamiques d'une application.
3	Implémenter un modèle de classes à l'aide d'un langage de programmation orientée objet.
4	Implémenter des tests unitaires pour vérifier de manière automatique le bon fonctionnement des classes et des méthodes.
5	Documenter systématiquement les classes d'une application et utiliser un générateur de documentation.

Domaine de compétence	Ingénierie d'applications
Objet	Applications avec 3-5 classes métiers (par ex. cartothèque, bibliothèque, gestion d'adresses, etc.)
Niveau	2
Prérequis	Compétences de base en programmation structurée.
Nombre de leçons	40
Reconnaissance	Certificat fédéral de capacité

Version du module	3.10

Connaissances opérationnelles nécessaires

Numéro de module	**226A**
Titre	Programmer selon les principes de la programmation orientée objet (sans héritage)

Compétence	Concevoir un modèle orienté objet à partir d'un cahier des charges qui ne nécessite pas l'utilisation des relations de réalisation ou de spécialisation/généralisation, implémenter et documenter les classes, puis les tester à l'aide d'outils adaptés.

Connaissances opérationnelles nécessaires

1.1 Connaître les moyens d'abstraction (relations, classes, attributs et méthodes) qui permettent de représenter le monde réel dans un modèle orienté objet.

1.2 Connaître la notion de classe en tant que types abstraits de données (*ADT*).

1.3 Connaître les relations d'association, d'agrégation et de composition.

1.4 Connaître les notions de classes métiers et de classes techniques.

1.5 Connaître les principes selon lesquels les objets d'une application interagissent (*few interfaces, small interfaces, explicit interfaces*) et appliquer le principe de délégation.

2.1 Connaître la notation UML pour les diagrammes de classes et d'objets (aspects statiques).

2.2 Connaître la notation UML pour les diagrammes de séquence et d'interaction (aspects dynamiques).

3.1 Connaître un langage de programmation qui supporte l'implémentation d'un modèle orienté objet à l'aide de classe (*direct mapping*).

3.2 Connaître et savoir appliquer les principes d'encapsulation et dissimulation d'information (*information hiding*) ainsi que le principe d'accès uniforme (*uniform access principle*) lors du développement de classes.

3.3 Connaître et comprendre la distinction entre la notion de classe qui représente le texte du programme, et celle d'objet créé à partir d'une classe durant l'exécution.

4.1 Connaître des procédures pour trouver des cas de tests permettant de vérifier le comportement avec des valeurs limites (*black box testing*) et les différents chemins d'exécution (*white box testing*).

4.2 Connaître les moyens mis à disposition par une infrastructure de tests unitaires (*unit testing framework*) pour l'implémentation des cas de test.

	4.3 Connaître des outils pour mesurer la couverture du code (*code coverage*). 5.1 Connaître le principe d'auto-documentation (self‑documentation principle) et les possibilités offertes par un générateur de documentation. 5.2 Connaître l'utilité de la documentation d'une application et savoir comment la générer.

Domaine de compétence	Ingénierie d'applications
Objet	Applications avec 3-5 classes métiers (par ex. cartothèque, bibliothèque, gestion d'adresses, etc.)
Niveau	2
Prérequis	Compétences de base en programmation structurée.
Nombre de leçons	40
Reconnaissance	Certificat fédéral de capacité

Version du module	3.10