Identification du module

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

Compétence	Concevoir un modèle orienté objet à partir d'un cahier des charges, 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 applicatives.
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 avancés pour des unités rendues indépendantes de certaines parties du système par des techniques appropriées.

Domaine de compétence	Ingénierie d'applications
Objet	Application avec 3-5 classes métiers (par ex. éditeur graphique, jeux simples, 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	**226B**
Titre	Programmer selon les principes de la programmation orientée objet (avec héritage)

Compétence	Concevoir un modèle orienté objet à partir d'un cahier des charges, implémenter et documenter les classes, puis les tester à l'aide d'outils adaptés.

Connaissances opérationnelles nécessaires

- 1.1 Connaître l'approche orientée objet avec classes et héritage.
- 1.2 Savoir appliquer l'héritage pour éliminer des redondances dans le modèle de classes.
- 1.3 Savoir appliquer le principe de délégation et l'héritage de façon à permettre l'extension d'une entité sans avoir à modifier son code source (*open/closed principle*), et faciliter ainsi la maintenance de l'application.
- 1.4 Connaître la notion de liaison dynamique (*dynamic binding*) ainsi que son utilité (*single choice principle*).
- 1.5 Savoir élaborer une hiérarchie de classes en utilisant les relations de spécialisation/généralisation et de réalisation (polymorphisme par sous-typage) ainsi que les notions de classe abstraite et d'interface.
- 2.1 Connaître la notation UML pour la relation de spécialisation/ généralisation (héritage)
- 2.2 Connaître la notation UML pour la relation de réalisation (implémentation d'interface)
- 3.1 Savoir implémenter la relation d'héritage à l'aide du langage de programmation étudié.
- 3.2 Savoir implémenter la surcharge de méthode (polymorphisme ad hoc) à l'aide du langage de programmation étudié.
- 3.3 Savoir implémenter la relation de réalisation d'interface à l'aide du langage de programmation étudié.
- 4.1 Savoir utiliser des objets factices (*mock objects*) pour rendre les tests unitaires indépendants de certaines parties inaccessibles ou non implémentées du système.
- 4.2 Connaître des moyens pour isoler les données de tests et les cas de tests afin d'assurer leur indépendance mutuelle.
- 4.3 Connaître des outils pour l'implémentation de cas de tests isolés.

Domaine de compétence	Ingénierie d'applications
Objet	Application avec 3-5 classes métiers (par ex. éditeur graphique, jeux simples, etc.).
Niveau	2
Prérequis	Compétences de base en programmation structurée.
Nombre de leçons	40
Reconnaissance	Certificat fédéral de capacité

Version du module	3.10