Programmation orientée-objet

Niveau d'étude	ECTS	Composante	Volume horaire	Période de l'année
Bac +3	**6 crédits**	**Sciences Fondamentales et Appliquées**	**50h**	**Semestre 5**

En bref

Langue(s) d'enseignement: Français
Méthode d'enseignement: En présence
Ouvert aux étudiants en échange: Oui

Présentation

Description

Programme résumé :

* éléments historique de la programmation orientée objet ;
* classes et objets : abstraction, encapsulation, constructeurs, visibilité ;
* associations de classes : agrégation, composition ;
* héritage et ses implications : redéfinition et surcharge de méthodes, contrôle de l'héritage, polymorphisme, sur-classement et sous-classement, classes abstraites et interfaces ;
* étude du langage Java et ses spécificités (utilisation d'un environnement de développement intégré, création et utilisation de bibliothèques, manipulation de collections, création et gestion des exceptions, manipulation de flux pour les entrées/sorties, sérialization, duplication d'objets (copie de surface et copie profonde), création et manipulation de threads ;
* gestion des tests : tests structurels, tests fonctionnels, critères de sélection de tests, diagrammes de flot de contrôle, types de tests, et mise en application avec l'utilisation de la bibliothèque JUnit ;
* UML : diagrammes de classes, diagrammes de séquences et diagrammes d'états.

Réalisation d'un projet à effectuer par petits groupes, où les étudiants doivent concevoir, spécifier, réaliser, tester et présenter leur application.

Objectifs

Connaître les fondements du paradigme de programmation orientée objet.

Heures d'enseignement

Programmation orientée-objet - CM	CM	18h
Programmation orientée-objet - TD	TD	14h
Programmation orientée-objet - TP	TP	18h

Pré-requis nécessaires

Niveau d'algorithmique et de programmation des 2 premières années de licence.

Compétences visées

* Être capable de mettre en oeuvre les mécanismes de bases de la programmation orientée-objets : abstraction, encapsulation, constructeurs, visibilité.
* Choisir les associations pertinentes entre classes : agrégation, composition.
* Être capable d'utiliser l'héritage et le polymorphisme à bon escient.
* Utiliser un environnement de développement intégré.
* Créer et manipuler des threads.
* Mettre en oeuvre et gérer des tests : tests structurels, tests fonctionnels, tests unitaires avec la bibliothèque JUnit ;
* Savoir concevoir une application orientée-objets avec le langage UML : diagrammes de classes, diagrammes de séquences et diagrammes d'états.

Infos pratiques

Lieu(x)

Futuroscope