

Java Objet

Présentation du cours

Olivier Cailloux

LAMSADE, Université Paris-Dauphine

Version du 8 janvier 2026

L'enseignant

- Olivier Cailloux ([coordonnées](#))
- Anciennement : Développeur sur projets de recherche
- Enseignant chercheur au LAMSADE

Objectifs pédagogiques

- Programmer de « vraies » applications
- De qualité
- Fournir et utiliser des composants réutilisables
- Moyens : interfaces claires, gestion des échecs, couplage faible, clarté conceptuelle du code, paradigme orienté objet
- Prise en main d'outils de dév. avancés :
 - Visual Studio Code
 - Maven
 - git

Contenu

- Syntaxe Java
- Programmation objets : responsabilités ; techniques
- Éléments d'ingénierie : programmation par contrat ; patrons de conception. . .
- Tests unitaires
- Collections
- Exceptions
- Utilisation de bibliothèques tierces
- Gestion des dépendances
- . . .

Intérêt pratique

- Technologies omniprésentes et très demandées (15 In-Demand Tech-Focused (And Tech-Adjacent) Skills And Specialties, 2022, [Forbes Technology Council](#) ; Emploi informatique 2023, [Developpez.com](#))
- Qu'on soit programmeur, qu'on discute avec des programmeurs (ou avec des robots !)
- Utile au-delà de la programmation
- Décomposition en responsabilités, en sous-problèmes
- Respect des spécifications

Demander simplement au chat ?

- Le chat doit savoir ce qui est à faire
- Comment savoir si le chat se trompe ? Lui demander d'expliquer ? Et si le chat **baratine** ?
- Le chat semble **aider** (pour le moment) surtout les débutants

Donc

- Savoir calculer, savoir coder
- Connaitre la plomberie sans faire de la plomberie
- Comprendre et critiquer

Prérequis

- Notions algorithmiques élémentaires (boucles, structures de listes, d'arbres...)
- Familiarité avec un langage tel que C++ ou Python
- Un ordinateur fonctionnel
- Manipulation de votre système d'exploitation : installation de logiciels, navigation dans le système de fichiers, démarrage de programmes
- Capacité à comprendre des textes en anglais liés à l'informatique

Évaluation

100% CC

- ≈ 4 tests en séance (annoncés)
- Devoirs maison : contributions individuelles à un projet commun
- Aggrégation (« owa ») des notes reçues au long de l'année
- Pondérations tests à déterminer ; exemple : 1 pour git et au mieux par individu parmi $\{1, 2, 3\}$
- Pondérations projet VS tests : au pire par individu parmi $\{1/5, 4/5\}$

Évaluation des tests

Avec ressources pré-existantes, sans IA générative

- Code doit compiler
- Livraison via git
- Respect précis des instructions
- Évaluation généralement automatique
- Points pour chaque aspect fonctionnel
- Rattrapage possible avec pénalité si solution presque fonctionnelle

Projet

IA génératives autorisées

- Projet commun : jeu de **localisation géographique** de pays
- Architecture client-serveur REST
- Outils modernes : Quarkus, Docker...
- À apprendre en autonomie !
- Bloquer deux dates de livraison, espacées d'au moins trois semaines
- Deux volontaires intégration (considérez le **biais de genre**)
- Livraison : indiquer sur devoir Moodle le lien **diff**

Travail attendu

- $(27,5 \text{ h} / \text{ECTS}) \times 4 \text{ ECTS}$
- $13 \times 3 \text{ h}$ en séance
- 24 h projet
- 4 heures de travail entre chaque séance en moyenne
- [Prenez des notes](#)
- Poursuivre les exercices chez vous, cf. page GitHub du cours

Licence

Cette présentation, et le code LaTeX associé, sont sous [licence MIT](#). Vous êtes libres de réutiliser des éléments de cette présentation, sous réserve de citer l'auteur.

Le travail réutilisé est à attribuer à [Olivier Cailloux](#), Université Paris-Dauphine.