Plan de cours

Numéro du cours	Pondération	Durée
420-FAF-LI	**2-5-3**	**105 heures**

Bloc 2

2022-06-20 – 2022-10-21

Titre du cours	Département
Programmation orientée objet II	**Informatique**

Cégep Limoilou
SERVICE AUX ENTREPRISES
FORMATION CONTINUE

Programme	Enseignant(e)s
AEC Développement d'applications sécuritaires *Cohorte 3*	**Eric Martel** *Contactez-moi par Teams ou MIO*

1. Modalités du cours

Le cours sera offert	En présence
Sur quelle plateforme de vidéoconférence les cours seront-ils donnés ?	Teams (si besoin)
Quels sont les applications ou logiciels à installer et à utiliser pour suivre le cours ?	Teams + voir section 6
À quel endroit retrouve-t-on la documentation relative aux cours (les notes de cours, capsules, exercices) ?	Teams
À quel endroit seront partagées les consignes pour les travaux et les évaluations?	Teams
Comment et quand rejoindre l'enseignant ?	Teams (préféré) / MIO Disponibilités : L'enseignant est complètement disponible lors des séances prévues à l'horaire et se rend également disponible autant que possible à l'extérieur des heures de cours.

Veuillez utiliser les liens suivants afin d'accéder aux présentations et aux procédures concernant l'utilisation de chacun des outils

Compte utilisateur du cégep	Installer Zoom	Installer Teams
Office 365	Antidote Web	Citrix : Accès aux laboratoires informatiques du cégep et accès aux machines virtuelles
Guide de l'étudiant	Étudier à distance	Liste complète des procédures

2. Présentation du cours

2.1 Place et rôle du cours dans le programme

Le cours ***420-FAF-LI : Programmation orientée objet II*** (POO II) situé au deuxième bloc est la suite du cours ***420-FAB-LI : Programmation orientée objet I.*** Ce cours donné simultanément au cours de ***420-FAH-LI :*** **Dév*eloppement informatique*** amène les étudiantes et étudiants à développer leurs compétences en lien avec divers concepts orientés objets (héritage, polymorphisme). Le cours ***420-FAH-LI : Développement informatique*** se donnant en parallèle permet d'approfondir certaines notions périphériques importantes pour un programmeur (gestion des sources, modélisation des concepts objets, validation de code avec les tests unitaires).

Dans ce cours, les étudiantes et étudiants approfondissent leurs connaissances de la programmation orientée objet en élaborant des programmes comprenant des liens hiérarchiques (classes, interfaces) et organiques (composition, agrégation, dépendance). Les étudiantes et étudiants toucheront également aux premières notions d'interfaces simples liés au code basé sur une approche MVC.

Ce cours se donne parallèlement à :

- ***420-FAG-LI - Programmation Web 2 :*** Programmation d'une application Web interactive en suivant les différentes étapes de conception préalables : *sitemap*, *sketch*, *wireframe*, *mockup* et *prototype*;
- ***420-FAH-LI - Développement informatique :*** Analyse des problèmes, débogage du code, application d'un plan de tests fonctionnel et production des diagrammes;
- ***420-FAE-LI - Bases de données 2*** *:* Création et utilisation des requêtes complexes, des fonctions, procédures et déclencheurs utilisant plusieurs tables afin d'exploiter une base de données de façon sécuritaire tout en assurant l'intégrité des données. Il permet d'apprendre et d'appliquer des concepts de base au niveau de l'administration d'une base de données (sauvegarde, réplication, gestion des utilisateurs.

Le cours ***420-FAB-LI – Programmation orientée objet 1*** est préalable relatif (à 50%) à ce cours.

2.2 Objectif terminal du cours

Au terme de ce cours, vous aurez programmé une application incluant des liens hiérarchiques et organiques entre les classes et composée d'une interface graphique.

L'atteinte de cet objectif implique que vous serez en mesure :

- d'établir des liens hiérarchiques et organiques entre différentes classes;
- de programmer des interfaces simples;
- d'utiliser des structures de données en mémoire.

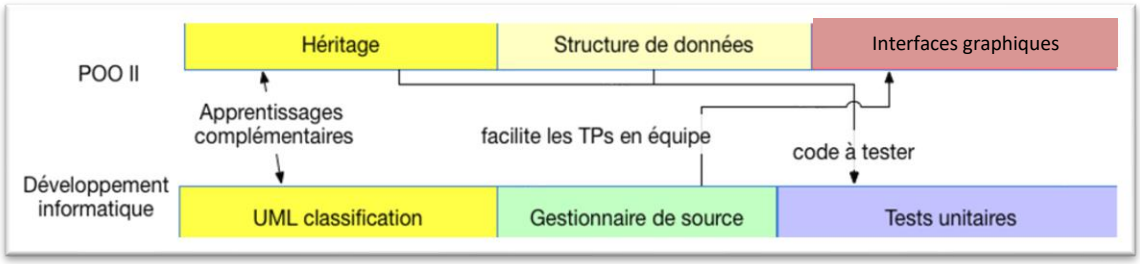
3. Organisation du cours

3.1 Contenu et apprentissages-clés

Capacités essentielles à développer ou à maîtriser	Savoir-penser nécessaires	Savoir-agir nécessaires	Savoir-devenir nécessaires
Capacité 1 : **Établir des liens hiérarchiques et organiques entre différentes classes**	• Établir des liens hiérarchiques entre les classes ; • Concevoir des algorithmes qui profitent du polymorphisme orienté objet ; • Déterminer lorsqu'un transtypage dynamique est nécessaire ; • Identifier les hiérarchisations multiples avec de l'héritage simple et des interfaces ; • Établir des liens organiques entre les classes ; • Identifier les substitutions possibles pour une classe.	• Programmer du code OO impliquant des classes hiérarchisées ; • Utiliser les chaînes de constructeurs ; • Utiliser le transtypage statique ou dynamique ; • Programmer des liens hiérarchiques et organiques ; • Utiliser les visibilités pour chaque élément ; • Programmer des algorithmes avec polymorphisme et transtypage dynamique ; • Utiliser l'encapsulation ; • Utiliser des types d'objets abstraits.	• Faire preuve de persévérance face aux apprentissages à intégrer et aux problèmes à résoudre ; • Développer de saines habitudes de travail en s'entraînant régulièrement et en exploitant l'éventail des ressources disponibles (documentation, références, Internet, professeur, etc.) ; • Faire preuve du souci du détail quant aux résultats attendus ; • Développer de bonnes habitudes en séparant de manière claire la vue de son modèle.
Capacité 2 : **Développer des programmes OO utilisant des structures de données dynamiques en mémoire**	• Connaitre quelques différentes structures de données dynamiques fournies par le langage (Listes, Dictionnaires, ...) ; • Choisir les structures de données appropriées pour la résolution d'un problème donné ; • Expliquer le fonctionnement des types génériques.	• Utiliser les types génériques avec les différentes collections ; • Utiliser des structures de données ; • Parcourir des structures de données selon plusieurs approches (boucle, itérateur).	
Capacité 3 : **Développer des programmes avec interface graphique**	• Identifier les éléments à afficher dans une interface graphique ; • Choisir les composants graphiques à mettre dans l'interface ; • Identifier les événements liés aux composants ; • Distinguer ce qui concerne le modèle de ce qui concerne la vue et son contrôleur.	• Utiliser des bibliothèques liées au développement d'interfaces ; • Utiliser d'autres composants graphiques (fenêtre, contrôle, boîte de dialogue, menu, etc.) pour affichage et saisie ; • Positionner des éléments graphiques de manière adéquate et cohérente ; • Gérer des événements liés aux interfaces (souris, clavier, fenêtre, etc.) ; • Déboguer le programme.	

3.2 Organisation du cours et méthodes pédagogiques

Ce cours se donne en parallèle au cours ***420-FAH-LI : Développement informatique***. Les compétences acquises en développement informatique sont importantes pour le cours POO II, car elles permettront d'utiliser des gestionnaires de sources, de mieux comprendre l'organisation du code grâce à une représentation graphique UML et finalement d'améliorer la qualité du code en utilisant des tests unitaires.



Le cours débute avec le développement de la capacité 1 (héritage et classification). La seconde partie du cours permet d'utiliser certaines structures de données. La dernière partie du cours aborde la création d'interfaces graphiques avec l'utilisation d'une ou plusieurs bibliothèques permettant la réalisation d'interfaces graphique. Une attention particulière est portée à la sensibilisation de l'étudiant au respect la séparation du modèle de la partie graphique (MVC et/ou MVVM). Cela permet de réaliser des programmes plus complets et de faire une synthèse sur les apprentissages en POO.

Les stratégies de formation suivantes peuvent être utilisées : animation d'ateliers théoriques, démonstration d'applications développées, exercices divers mettant en pratique les notions récemment apprises, consultation de l'aide en ligne, réalisation d'applications de plus en plus complexes qui respectent des normes de qualité et de sécurité.

3.3 Calendrier des activités[1]

Semaine	Théorie	Exercices et évaluations
01 **06-20 (8h)**	• Introduction à la programmation d'applications à interfaces graphiques (WPF au début, puis ASP .NET MVC en cours de bloc)	Exercices
02 **06-27 (7,5h)**	• Surcharge de méthodes	Défi 1
03 **07-04 (7,5h)**	• Structures de données unidimensionnelles en C# (listes, files, piles, dictionnaires, listes chaînées…) et méthodes associées (tri, copie, fusion…)	Défi 2
04 **07-11 (7,5h)**	• Structures de données multidimensionnelles en C# (tableaux multidimensionnels et tableaux en escalier)	Défi 3
05 **07-18 (0h)**	**Vacances**	
06 **07-25 (0h)**	**Vacances**	
07 **08-01 (0h)**	**Vacances**	
08 **08-08 (7,5h)**	• Héritage (incluant substitution, abstraction et blocage)	Défi 4
09 **08-15 (7,5h)**	• Interfaces (POO)	Défi 5 / **Date limite défi 1**
10 **08-22 (7,5h)**	• Accès et sauvegarde de données	Défi 6 / **Date limite défi 2**
11 **08-29 (7,5h)**	• Surcharge d'opérateurs et indexeur	Défi 7 / **Date limite défi 3**
12 **09-05 (3,5h)**	• Génériques	Défi 8 / **Date limite défi 4**
13 **09-12 (7,5h)**	• Itérateurs	Défi 9 / **Date limite défi 5**
14 **09-19 (7,5h)**	Temps pour terminer les défis	Défi 10 / **Date limite défi 6**
15 **09-26 (7,5h)**	Temps pour terminer les défis	**Date limite défi 7**
16 **10-03 (7,5h)**	Temps pour terminer les défis	**Date limite défi 8**
17 **10-10 (3,5h)**	Temps pour terminer les défis	**Date limite défi 9**
18 **10-17 (7,5h)**	Temps pour terminer les défis	**Date limite défi 10**

[1] *Cet échéancier est donné à titre indicatif. L'enseignant peut modifier l'ordre de présentation et les délais prévus pour chaque activité. Si un événement quelconque empêche la tenue d'un examen, celui-ci est automatiquement remis au cours suivant.*

4. Modalités d'évaluation des apprentissages

Il n'y aura aucun examen dans ce cours. Vous aurez des défis à relever tout au long de la session, et vous devrez démontrer à votre enseignant votre solution ainsi que votre maîtrise de cette dernière pour chaque défi au fur et à mesure. Une remise par semaine est en principe attendue, mais vous aurez quatre semaines pour terminer chaque défi. Vous aurez droit de présenter le même défi à plusieurs reprises afin d'améliorer votre note. Les défis seront en lien général avec certaines notions attendues selon le calendrier, mais comme les défis seront eux-mêmes de nature formative, votre autonomie sera très sollicitée.

4.1 Liste des évaluations (formatives et sommatives)

Les divers travaux qui sont utilisés pour l'évaluation sommative sont listés dans le tableau ci-après décrivant les éléments évalués dans chacun.

Répartition des évaluations par défi et par capacité

	Défi 1	Défi 2	Défi 3	Défi 4	Défi 5	Défi 6	Défi 7	Défi 8	Défi 9	Défi 10	Total
Capacité 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**40 %**
Capacité 2		✓	✓	✓	✓	✓	✓	✓	✓	✓	**30 %**
Capacité 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**30 %**
Pondération	**6 %**	**10 %**	**10 %**	**10 %**	**10 %**	**10 %**	**10 %**	**10 %**	**10 %**	**14 %**	**100 %**

Important : *Une note de 0 % sera donnée à un défi en cas de plagiat pour une première fois, et une note de 0 % sera donnée pour le cours au complet en cas de récidive.*

4.2 Capacités et aspects/critères d'évaluation

Capacités essentielles à évaluer sommativement	Aspects évalués et critères d'évaluation		
	Cible (Niveau optimal)	**Seuil de réussite (Niveau minimal)**	**Moyens (dispositifs) privilégiés**
Capacité 1 : **Établir des liens hiérarchiques et organiques entre différentes classes** **Pondération : 40%**	• Programmer du code OO **fonctionnel**, **optimisé** et bien **documenté** ; • Utiliser **efficacement et systématiquement** les chaînes de constructeurs ; • Utiliser **systématiquement** le transtypage **approprié** statique ou dynamique ; • Programmer **continuellement** des liens hiérarchiques et organiques **pertinents** ; • Utiliser **continuellement** les visibilités adéquates pour chaque élément ; • Programmer des algorithmes **efficaces** avec polymorphisme et transtypage dynamique ; • Utiliser l'encapsulation **au maximum** ; • Utiliser **continuellement** le type d'objet le plus abstrait possible.	• Programmer du code OO **fonctionnel** (qui compile sans erreurs de syntaxe mais pouvant comporter des erreurs de logique); • Utiliser **efficacement et systématiquement** les chaînes de constructeurs ; • Utiliser **systématiquement** le transtypage **approprié** statique ou dynamique ; • Programmer **dans la plupart des cas** des liens hiérarchiques et organiques **pertinents** ; • Utiliser **dans la plupart des cas** les visibilités adéquates pour chaque élément ; • Programmer des algorithmes **fonctionnels** avec polymorphisme et transtypage dynamique ; • Utiliser l'encapsulation **minimalement** ; • Utiliser **minimalement** des types d'objets abstraits.	• Travaux pratiques en laboratoire - en environnement non contrôlé et à au moins 2 reprises • Examens pratiques – individuels, en environnement contrôlé et à 2 reprises
Capacité 2 : **Développer des programmes OO utilisant des structures de données en mémoire** **Pondération : 30%**	• Utiliser **continuellement** les types génériques avec les différentes collections ; • Développer des algorithmes **efficaces** permettant de parcourir des structures de données (boucle, itérateurs, flux).	• Utiliser **régulièrement** les types génériques avec les différentes collections ; • Développer des algorithmes permettant de **parcourir minimalement** des structures de données (boucle, itérateurs, flux).	• Examens (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative. • Laboratoires (individuels) – en environnement non contrôlé et sont évalués à au moins 2 reprises de façon sommative. •
Capacité 3 : **Développer des programmes avec interface graphique** **Pondération : 30%**	• Choisir et utiliser **continuellement** des éléments graphiques pour l'affichage et la saisie de manière **appropriée;** • Gérer de manière **exhaustive** les événements liés aux interfaces. • Séparer **systématiquement** le modèle de ce qui concerne la vue de manière appropriée • Positionner **systématiquement** les composants **de manière adéquate et pertinente** dans l'interface.	• Choisir et utiliser **dans la plupart des cas** des éléments graphiques pour l'affichage et la saisie de manière **fonctionnelle ;** • Gérer **sommairement ou avec quelques erreurs** les événements liés aux interfaces. • Séparer **la plupart du temps** le modèle de ce qui concerne la vue **avec quelques erreurs.** • Positionner **la plupart du temps** les composants **de manière fonctionnelle** dans l'interface.	• Examens (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative. • Laboratoires (individuels) – en environnement non contrôlé et sont évalués à au moins 2 reprises de façon sommative.

4.4 Conditions de réussite du cours

Pour réussir le cours, vous devrez avoir démontré avoir atteint le seuil de réussite de chacune des trois capacités à au moins 60 %.
Le processus et le produit doivent être évalués au moins deux fois de manière sommative pour chaque capacité. Une évaluation peut regrouper l'évaluation de plusieurs capacités.

4.5 Modalités de reprise d'une évaluation sommative

- Sur présentation d'une attestation écrite officielle, si vous êtes absent pour cause de force majeure (maladie, accident, décès d'un proche parent, subpoena, etc.) vous pourrez passer une évaluation à une date fixée conjointement avec l'enseignant concerné.
- Les voyages, vacances et activités de loisir ne constituent pas des cas de force majeure.
- L'absence à un cours ne relevant pas d'un cas de force majeure ne peut justifier la remise en retard d'un travail exigé par l'enseignant.

5. Coordonnées et disponibilités de l'enseignant

Enseignant : Eric Martel
Communication : Par MS-Teams préférentiellement, sinon par Mio
Disponibilités : L'enseignant est complètement disponible lors des séances prévues à l'horaire et se rend également disponible autant que possible à tout moment à l'extérieur des heures de cours.

6. Matériel

Obligatoire :

- La méthode d'accès aux notes de cours ainsi que les capsules vidéo seront disponibles sur Teams.

Logiciel utilisé dans le cours :

- Environnement de développement intégré Microsoft Visual Studio

Vous pouvez utiliser :

- un ordinateur que vous possédez déjà (poste fixe ou portable) pourvu qu'il possède les caractéristiques minimales nécessaires;
- un des ordinateurs proposés dans le cadre de l'entente avec la Coop Zone;
- un autre modèle que vous achetez chez un autre fournisseur et qui possède les caractéristiques minimales nécessaires.

Les caractéristiques minimales recommandées pour un ordinateur portable avec un système d'exploitation PC (Windows 10) sont :

- une carte réseau WiFi;
- un processeur i5;
- une mémoire de 8 Go (16 Go recommandée);
- un disque dur SSD de 250 Go;
- un écran de 14 ou 15 pouces;
- une carte vidéo dédiée;
- une caméra web (intégrée ou externe);
- un micro (intégré ou externe);

- deux (2) ports USB.

Voici également les accessoires optionnels que nous vous recommandons de vous procurer :

- Une mallette de transport ou sac à dos conçu à cet effet;
- Un câble de sécurité;
- Un casque d'écoute avec micro;
- Une souris optique (avec ou sans fil);
- Une clé de mémoire USB (minimum 16 Go).

Pour vous procurer un ordinateur dans le cadre de l'entente avec la coop Zone :
Vous pouvez vous procurer l'ordinateur par un achat en ligne en cliquant sur le lien suivant : https://www.zone.coop/informatique/programmes-dacquisition-dordinateur/limoilou.html

Vous pouvez aussi vous rendre aux différents points de vente de la coop. Pour les heures d'ouverture, nous vous invitons à consulter la page suivante: https://www.zone.coop/la-cooperative/heures-d-ouverture.html

7. Médiagraphie

- Microsoft Corporation. (2021, 14 mai). *Documentation C#.* https://docs.microsoft.com/fr-fr/dotnet/csharp/
- Hugon, Jérôme (2019). *C# 8 - Développez des applications Windows avec Visual Studio 2019.* Éditions ENI.

10. Code de vie

Les enseignantes et enseignants du Département d'informatique ont adopté les éléments suivants dans une optique de favoriser la réussite dans les cours.

Représentant de classe
Un représentant de classe pourrait être désigné dans les deux premières semaines de la session afin de créer une synergie dans les groupes et d'avoir une personne identifiée pour aider à l'amélioration continue des cours. Deux rencontres pourraient être faites par session avec ces représentants afin de connaître les doléances, s'il y a lieu.

Présence active
Nous considérons que la présence active en classe est essentielle à la réussite scolaire. En ce sens, le département s'attend à ce que les étudiants assistent à tous les cours et fassent tous les travaux formatifs demandés par leurs enseignants.

> Par présence active, nous entendons plus qu'une simple présence physique en classe. La présence active comprend également la participation aux activités d'apprentissage (incluant les lectures préalables), la réalisation des travaux et exercices formatifs suggérés, la réalisation des travaux et exercices sommatifs demandés et enfin, l'implication personnelle dans les travaux d'équipes.

Absence
Les enseignants prennent les présences à chacun des cours. Une étudiante ou un étudiant qui s'absente a la responsabilité de :

- Contacter son enseignant dans les plus brefs délais afin de déterminer une méthode de régulation de la situation avec de son enseignant (ex : travaux et lecture);
- Faire un suivi des mesures préconisées auprès de son enseignant (Vérifier les travaux, discussion avec son enseignant, etc.).

Plagiat

Le plagiat est formellement interdit, sous quelque forme que ce soit.

Est notamment considéré comme du plagiat le fait de reproduire ou de réutiliser en tout ou en partie le travail d'un autre étudiant, avec ou sans son autorisation; dans le cas où il y aurait eu autorisation, l'étudiant plagié sera également considéré coupable de plagiat par complicité.

La politique du cégep[2] stipule qu'une étudiante ou un étudiant reconnu coupable de plagiat se voit octroyer la note de zéro lors d'un premier acte, puis la mention « échec » au cours lors d'un deuxième acte. Advenant un troisième acte de plagiat, la sanction pourra aller jusqu'au renvoi.

Ressources, conseils et aide-mémoire : https://www.cegeplimoilou.ca/etudiants/carrefour-de-l-information/bibliotheques/guides/presenter-un-travail-ecrit/antiplagiat/

[2] https://www.cegeplimoilou.ca/media/1291293/b_07-politique-institutionnelle-d-evaluation-des-apprentissages-piea_fev2016.pdf, Annexe 1

9. Modalités départementales d'évaluation des apprentissages[3]

En conformité avec la politique institutionnelle d'évaluation des apprentissages (P.I.É.A) version 2014, le département d'informatique apporte les précisions suivantes :

Article 6.6 *L'évaluation de l'expression et de la communication en français*	Les exigences minimales de qualité des écrits produits par l'étudiante ou l'étudiant sont établies selon les aspects et les critères suivants, permettant essentiellement une communication suffisamment claire pour que l'enseignante ou l'enseignant puisse poser un jugement sur la matière faisant l'objet de l'évaluation : • L'organisation et l'expression des idées sont suffisamment logiques et cohérentes pour permettre la compréhension aisée de l'écrit; • La syntaxe et la ponctuation rendent la lecture aisée malgré des faiblesses évidentes ; • Les fautes d'orthographe (d'usage et grammaticale) sont fréquentes, mais ne nuisent pas à la compréhension de l'écrit ; • Le vocabulaire, notamment la terminologie propre au domaine, est sommaire, mais utilisé de manière juste et adéquate. Toute évaluation sommative peut être refusée par l'enseignante ou l'enseignant si elle ne répond pas aux exigences minimales de qualité de la langue exprimées précédemment. L'enseignante ou l'enseignant accorde à l'étudiante ou l'étudiant un délai de reprise de 24 h pour la reformulation du texte, délai dont le début est à la convenance des deux parties.
Article 12.1 *L'équivalence de l'évaluation des apprentissages*	Lorsqu'un cours est donné par plusieurs enseignants, un comité de cours est formé afin d'établir le plan de cours, la valeur attribuée à chaque objet d'évaluation, les tâches servant à l'évaluation sommative et leur contexte de réalisation et des grilles d'évaluations afin d'assurer une uniformité dans la correction. Ce comité doit se réunir au besoin pendant la session afin de s'assurer de l'uniformité du contenu enseigné et des modalités de correction.
Article 12.3 *La remise en retard de réalisations servant à l'évaluation sommative des apprentissages*	À l'exception des cas de force majeure, une remise en retard des travaux entraîne automatiquement une pénalité de 10 % par jour de retard. *Une fois qu'une réalisation servant à l'évaluation est corrigée et remise aux étudiants, l'étudiant qui n'a pas encore remis son travail se voit attribuer la note zéro à cette activité d'évaluation.*
Article 13.1 *Le réexamen d'un résultat attribué pour une tâche*	L'étudiant qui veut faire une demande de révision de notes partielle doit d'abord s'adresser à son enseignant ou enseignante. S'il est insatisfait du résultat de sa démarche, il s'adresse alors au coordonnateur du département. Celui-ci dirigera un comité formé de l'enseignant concerné et d'un enseignant qui possède les compétences à juger les motifs du litige. Le résultat devra être communiqué à l'étudiant dans les 10 jours ouvrables suivant sa demande.

[3] *Adopté par le département d'informatique, le 14 mars 2016*