PLAN DE COURS

Sigle : **420-KBB-LG**

Titre : **Programmation orientée objet avancée**

Programme : **Techniques de l'informatique (420.B0)**

Discipline : **Informatique**

Secteur : **Enseignement régulier**

Pondération : **3-3-2**

Préalable : **420-KB4-LG**

Session : **Automne 2025**

Professeur	Bureau	Courriel	Téléphone (450) 430-3120
Marc Beaulne	F-322	marc.beaulne@clg.qc.ca	poste 2369
Patrice Roy	F-314	patrice.roy@clg.qc.ca	poste 2780
Coordonnateurs	**Bureau**	**Courriel**	**Téléphone**
Martin Lemay	F-320	Martin.Lemay@clg.qc.ca	s/o
François Simard	F-312	Francois.Simard@clg.qc.ca	s/o

Présentation du cours

Ce cours a pour but de solidifier la maîtrise des notions de la programmation orientée objet vues précédemment dans le programme, de même que d'amener l'étudiante ou l'étudiant à se familiariser avec la multiprogrammation, les schémas de conception, la programmation générique et la programmation par événements.

Contribution du cours au programme

La branche Programmation sert à titre d'axe fondamental et porteur du programme Techniques de l'informatique, et ce cours est le dernier de cette branche. Le cours 420-KBB permettra à l'étudiante ou à l'étudiant de parfaire et de consolider ses compétences en programmation. Ces compétences seront réinvesties dans le développement d'applications Web, d'applications mobiles, de jeux vidéo ou d'applications utilisées dans des environnements intelligents.

Compétence du devis ministériel

Ce cours poursuit et termine l'acquisition de la compétence 00Q6 – Exploiter les principes de la programmation orientée objet, dont la formulation se trouve en annexe à la fin de ce plan de cours.

Charge de travail

La pondération de ce cours est 3-3-2. Cela signifie qu'en plus des trois heures de cours théoriques et des trois heures de cours pratiques, l'étudiante ou l'étudiant doit s'attendre à fournir, **en moyenne**, deux heures de travail personnel par semaine à l'extérieur du cours.

Objectif d'intégration

À la fin de ce cours, l'étudiant ou l'étudiante sera en mesure de créer une application en utilisant la programmation orientée objet avancée et les idiomatiques propres à un langage de programmation, à partir de consignes et d'une description des fonctionnalités de l'application.

Outils

Ce cours utilisera le langage de programmation C# et l'environnement de développement Visual Studio dans sa déclinaison la plus récente. Prenez soin de mettre Visual Studio à jour dès le début de la session, pour éviter des problèmes de compatibilité de versions lors des travaux pratiques.

Objectifs d'apprentissage

L'objectif général du cours est d'exploiter les concepts avancés de la programmation orientée objet en C# pour créer des applications. Une exploration plus détaillée de cet objectif suit; notez que les éléments listés ne seront pas nécessairement couverts dans l'ordre selon lequel ils sont présentés.

Le cours vise le développement par l'étudiante ou par l'étudiant de la rigueur et de l'esprit d'analyse, plus particulièrement en ce qui a trait à qualité de la programmation, à la robustesse et à l'auto-documentation du code, la conception de classes, l'application de schémas de conception, la gestion des états et le développement d'algorithmes.

Évaluation

L'évaluation des apprentissages se fera à l'aide :

- de plusieurs exercices et laboratoires qui seront des travaux à faire sur une base quasi hebdomadaire;
- d'au moins huit minitests qui sont des évaluations théoriques courtes, distribués sans préavis en début de cours, dont nous ne retiendrons que les six meilleurs résultats; et
- d'une production finale d'intégration composée d'une partie théorique (un examen synthèse) et d'une partie pratique (un travail de fin de session).

Le ou vers le[1]	**Type d'évaluation**	**Pondération**
À partir de la semaine du 21 nov.	Activité intégratrice pratique (PFI)	20
Semaine du 2 décembre	Activité intégratrice théorique (examen final)	20
Durant la session	Exercices et laboratoires	30
Durant la session	Minitests	30
Total :		**100**

[1] Ces dates sont sujettes à ajustements selon la vitesse du groupe et autre imprévue.

Production finale d'intégration

La production finale d'intégration comporte un volet théorique et un volet pratique. La pondération du volet pratique est plus petite ou égale à la pondération du volet théorique et le poids total des deux évaluations est conforme aux critères de la PIEA.

Plus précisément, le volet pratique de la production finale d'intégration vise à vérifier la capacité individuelle de l'étudiante ou de l'étudiant à compléter le code d'une application informatique relativement complexe en y ajoutant les parties qui apportent une solution algorithmique aux sous-problèmes plus simples à résoudre, le tout en mettant à profit les acquis de ce cours.

Par ailleurs, le volet théorique de la production finale d'intégration vise à vérifier la compréhension théorique des concepts enseignés durant toute la session et la capacité individuelle de l'étudiante ou de l'étudiant à écrire des algorithmes et des classes pour résoudre des problèmes simples et les coder sans l'aide de l'ordinateur. Ce volet de la production finale pourra, par exemple, prendre la forme d'un examen écrit.

Critères d'évaluation de la production finale d'intégration

Pour la production théorique :

- qualité de la compréhension des concepts théoriques;
- précision des réponses;
- qualité de la présentation écrite.

Pour la production pratique :

- qualité de l'exécution du programme;
- précision de l'application produite relativement au problème donné;
- qualité de la codification et de la conception.

Règles régissant l'évaluation

1. Sauf circonstances exceptionnelles dont le professeur est seul juge, l'absence à un examen entraîne la note zéro pour cet examen. Il n'y a pas d'examen de reprise. Si le professeur juge qu'il y a des circonstances exceptionnelles, il pourra proposer à l'étudiant un arrangement individuel ou encore imposera des conditions de réussite particulières. Chaque cas sera analysé au mérite.

   C'est à l'étudiante ou à l'étudiant qu'il revient de rencontrer son enseignante ou son enseignant pour lui faire part, dès son retour au Collège, des motifs de son absence et lui fournir une pièce justificative. Dans le cas d'un motif exceptionnel et justifié, l'enseignante ou l'enseignant pourra proposer à l'étudiante ou à l'étudiant une modalité de reprise ou une modification de barème.

2. Sauf circonstances exceptionnelles dont le professeur est seul juge, tout travail en retard **ne sera pas corrigé** et aura droit à la note **zéro**.

   La politique institutionnelle d'évaluation des apprentissages (PIEA) prévoit, à la section 7.6 PLAGIAT ET TRICHERIE, des dispositions en cas de plagiat ou de tricherie. Nous vous invitons fortement à en prendre connaissance.

3. La réussite du cours est conditionnelle à la réussite de l'activité synthèse. Le seuil de réussite est fixé à 60 % pour cette activité. Vous devez avoir au moins 60% dans le volet théorique et au moins 60% dans le volet pratique sinon vous aurez une note d'échec à ce cours.

4. Un travail peut voir sa note varier jusqu'à 10 % en plus ou en moins en raison de l'excellence ou la déficience du français lorsqu'applicable.

Matériel obligatoire à se procurer

Aucun matériel obligatoire n'est prévu pour ce cours. Toutefois, l'étudiante ou l'étudiant est responsable de prendre les notes dont elle ou il a besoin pendant les séances en classe.

Contenu

Le contenu suivant sera abordé.

Sur le plan structurel

- Distinguer « types valeurs » et « types références » dans leur acception C#. En particulier, distinguer les mots-clés `struct` et `class`
- Distinguer les paramètres `in`, `out` et `ref`.
- Comprendre le *Boxing* et le *Unboxing*.

Sur le plan algorithmique

- Se familiariser avec la multiprogrammation à l'aide de multiples fils d'exécution et à l'aide de méthodes asynchrones (coroutines).
- Se familiariser avec les délégués.
- Concevoir des classes et des méthodes génériques avec contraintes.
- Exprimer un algorithme sous forme générique.
- Se familiariser avec les pratiques associées à LiNQ.
- Utiliser des expressions λ.
- Sérialiser et désérialiser un objet.

Sur le plan programmatique

- Mesurer et optimiser le temps d'exécution.
- Se familiariser avec la gestion des ressources.
- Se familiariser avec la couverture de code dans les tests.
- Utiliser un profileur.
- Utiliser les uplets et les dictionnaires.

Sur le plan architectural

- Créer une bibliothèque à liens dynamiques.
- Structurer une application orientée objet.
- Introduction aux schémas de conception : Observateur, Fabrique (*Factory*), Singleton
- Se familiariser avec les machines à états finis et les graphes.
- Se familiariser avec le développement dirigé par les données (*Data-Driven*).

Méthodologie

Les séances se passent toutes au laboratoire. Ceci nous permet d'alterner exposés magistraux, études de cas et démonstration à l'écran ainsi que séances d'exercices.

Le système de développement *Visual Studio 2022* servira d'outil pour les applications pratiques. Nous utiliserons le langage C# de Microsoft comme outil de développement. Les étudiantes et les étudiants peuvent télécharger gratuitement *Visual Studio 2022* à partir du site de Microsoft.

Il est également possible d'utiliser https://dotnetfiddle.net/ si vous utilisez une machine sur laquelle *Visual Studio 2022* n'est pas installée, du moins pour résoudre des problèmes simples, ou simplement parce que vous souhaitez tester vos idées sans charger un outil plus lourd.

Médiagraphie

Les documents requis durant le cours, qu'il s'agisse des notes de cours théoriques, des laboratoires ou des exemples, seront disponibles par Colnet dans l'onglet **Ressources** de la rubrique **Cours**. Votre enseignant pourra toutefois décider de procéder autrement et passer via son propre site Web ou d'une autre manière.

Le site https://h-deb.ca est également un site de référence intéressant pour le cours. Il s'agit d'un site généraliste sur la programmation en langue française.

Démarche officielle du Cégep concernant les conflits enseignants / étudiants

Référence : http://www.clg.qc.ca/cheminement-registrariat/conflit-entre-un-enseignant-et-un-etudiant/

En cas de conflit avec un enseignant, il est important de suivre la démarche suivante :

1. Dans un premier temps, il faut en parler directement avec l'enseignant concerné et miser sur le dialogue
2. Dans le cas où le différend ne se règle pas, l'étudiant s'adresse au coordonnateur du département concerné et tente de régler le problème avec son aide
3. Si le problème persiste toujours, l'étudiant peut rencontrer son API au local L-117, qui analysera la situation problématique avec lui et qui pourra lui suggérer de déposer une plainte en remplissant le formulaire à cet effet. La plainte sera acheminée au directeur adjoint de la Direction des études qui verra à trouver une solution

Annexe A – Compétence 00Q6

Énoncé de la compétence	Contexte de réalisation
00Q6 Exploiter les principes de la programmation orientée objet	À partir d'un problème. À l'aide de règles de nomenclature et de codage.

Éléments de la compétence	Critères de performance
1. Analyser le problème.	• Décomposition du problème en fonction des exigences de l'approche orientée objet. • Détermination correcte des données d'entrée, des données de sortie et de la nature des traitements. • Détermination juste des classes à modéliser. • Détermination correcte des algorithmes à produire.
2. Modéliser les classes.	• Détermination correcte des attributs et des méthodes des classes. • Application judicieuse des principes d'encapsulation et d'héritage. • Représentation graphique correcte des classes et de leurs relations. • Respect des règles de nomenclature.
3. Produire les algorithmes pour les méthodes.	• Détermination adéquate des opérations nécessaires pour chaque méthode. • Détermination correcte d'une séquence logique des opérations. • Vérification appropriée du fonctionnement des algorithmes. • Représentation correcte des algorithmes.
4. Générer l'interface graphique.	• Choix approprié des éléments graphiques pour l'affichage et la saisie. • Positionnement correct des éléments graphiques. • Paramétrage correct des éléments graphiques.
5. Programmer des classes.	• Choix approprié des instructions, des types de données élémentaires et des structures de données. • Organisation logique des instructions. • Programmation correcte des messages à afficher à l'utilisatrice ou à l'utilisateur. • Intégration correcte des classes dans le programme. • Fonctionnement correct du programme. • Respect de la syntaxe du langage. • Respect des règles de codage.

6. Documenter la programmation.	• Notation claire de commentaires dans le code informatique. • Notation claire de la documentation d'aide à la programmation. • Utilisation appropriée des générateurs de documentation.
7. Appliquer la procédure liée à la gestion des versions de programmes.	• Configuration correcte du système de gestion de versions. • Soumission méthodique du code modifié. • Gestion judicieuse des branches et des conflits.