UNIVERSITÉ DE SHERBROOKE

Faculté des sciences
Centre de formation en technologies de l'information

Plan de cours

Cours :	INF-731, Programmation orientée objet
Trimestre :	Automne 2020
Enseignant :	Pierre Prud'homme

1. Contexte du cours

La programmation orientée objet, bien que datant en principe des années '70, est devenue au cours des années '90 le paradigme dominant du développement de systèmes. Bien qu'on observe une résurgence des langages de script dans la programmation Web (PHP[1] et JavaScript sont de bons exemples), les langages orientés objet ont une approche qui les rend plus aptes à la conception rapide de systèmes complexes; la philosophie derrière les applications les plus vendues est orientée objet; l'analyse orientée objet a pris une place importante dans les façons de faire.

Comprendre les principes fondamentaux de la programmation orientée objet est devenu un passage nécessaire pour agir dans le monde du développement en entreprise. Dans votre futur rôle, diriger les professionnels du développement fera partie de votre description de tâche. La mise en pratique que vous propose ce cours vous permettra de mieux comprendre les considérations auxquelles ils doivent faire face. Ce sera un atout significatif qui contribuera à établir votre crédibilité en entreprise.

2. Place du cours dans le programme

Ce cours présume qu'a priori, l'étudiant(e) comprend et sait utiliser:

- les concepts de programmation que sont la ***séquence***, ***l'alternative*** et la ***répétitive***;
- les concepts de ***variable*** et de ***constante***;
- les concepts de ***procédure*** et de ***fonction***, y compris la notion intimement liée de ***paramètre par valeur*** et ***par référence***;
- les ***types de données***, au moins les types *primitifs*, les *enregistrements* et les *tableaux*.

L'étudiant(e) est considéré avoir une connaissance suffisante de la syntaxe du langage C# qui sera notre outil de développement dans le cadre du cours.

L'approche objet étant le modèle dominant à plusieurs égards dans l'industrie du développement logiciel et de l'analyse contemporains, ce cours visera à assurer l'acquisition par l'étudiant(e) d'un bagage préalable aux cours du programme impliquant une part de développement logiciel. Le cours est normalement préalable au cours *Applications Internet*

[1] 'Langage de script' n'exclut pas la notion d'orienté objet. Le support objet a été amélioré en PHP 5 et il existe des outils qui intègrent la philosophie objet à PHP.

(INF 777) et aidera à mieux saisir les notions vues dans les cours *Conception et évaluation d'IPM* (INF 753) et, dans une moindre mesure, *Méthodes d'analyse et de conception* (INF 755).

3. Objectifs généraux du cours

Ce cours vise à donner à l'étudiant(e) une *connaissance appliquée* de la programmation et de la philosophie orientée objet.

Connaissance appliquée signifiera à la fois le *savoir* permettant de comprendre le fonctionnement des systèmes développés selon les principes du modèle orienté objet, et également le *savoir-faire* requis pour en tirer soi-même profit dans un contexte de développement.

Afin d'atteindre cet objectif, nous développerons des **habiletés de programmation** en utilisant un langage facilitant la mise en pratique des principes exposés. Le langage de programmation retenu pour ce cours est le C#.

Parenthèse

Le modèle objet est un incontournable de presque tout développement informatique en industrie aujourd'hui. La pensée objet, en informatique, sauve du temps et de l'argent.

Il s'agit d'une manière d'aborder les problèmes avec laquelle vos employés versés sur le plan technique seront familiers et que vous devrez comprendre *suffisamment* pour être en mesure d'en discuter avec eux de manière crédible afin de gagner leur respect.

Comme pour la plupart des modes de pensée, bien comprendre le modèle objet demande de l'avoir suffisamment appliqué pour en saisir les principales considérations — se limiter à un simple survol est un bon moyen de mal paraître. C'est pourquoi nous allons apprendre ici *comment programmer* selon le modèle orienté objet, avec un langage largement répandu, de manière à en saisir *le plus possible (mais quand même partiellement)* les enjeux et subtilités. Notre but n'est toutefois pas de vous permettre de développer une expertise suffisante pour remplacer les membres de votre équipe de développement.

Contenu

Historique et fondements du modèle OO. Bases du C#. Classes et instances. Constructeur, destructeur. Méthodes, attributs. Encapsulation. Surcharge d'opérateurs. Héritage simple. Gestion de la mémoire. Abstraction. Méthodes virtuelles et polymorphismes. Considérations de design. Mise en pratique des concepts dans le cadre de travaux de développement.

4. Objectifs spécifiques

Au terme du cours, l'étudiant(e) sera capable de:

{1} établir de quelle manière le problème à résoudre se prête à une solution orientée objet;
{2} analyser un problème pour développer une solution orientée objet;
{3} rédiger un programme orienté objet résolvant un problème concret;
{4} respecter des standards de programmation;
{5} appliquer correctement les grands principes du modèle objet;
{6} produire une hiérarchie de classes et considérer la performance;
{7} généraliser le comportement d'une hiérarchie de classes par polymorphisme;
{8} utiliser des collections et appliquer des éléments de programmation générique;
{9} appliquer le modèle objet à un problème de développement logiciel ou matériel impliquant plusieurs intervenants.

5. Organisation du cours

Le cours sera organisé sous forme de courts chapitres d'une ou deux séances environ, énumérés ci-après dans l'ordre où ils seront couverts. Pour chaque thème, le livre suggéré du cours fournira un complément de théorie. En plus du livre de référence, l'enseignant fournira d'autres documents d'appoint pour couvrir certains concepts.

Planification des séances hebdomadaires

Séance(s)	**Contenu prévu du cours**
S1	**Entrée en matière** Présentation du professeur, du plan de cours, de l'approche retenue pour le cours. Place du cours dans votre programme. Formule utilisée pour l'évaluation. **Rappel de notions supposées connues de l'outil de programmation** Rappel des grandes lignes de la syntaxe du langage C# sur le plan que l'on pourrait qualifier de structuré. Le rythme de la séance sera soutenu puisqu'il est présumé que les étudiant(e)s ont tous et toutes la base de programmation requise et qu'il s'agit pour l'essentiel d'un rappel. On portera une attention particulière sur la distinction fondamentale à faire entre **type valeur** et **type référence** dans le langage. **Formule pédagogique :** cette séance aura lieu en classe. **Référence(s)**: **[P1]** chapitres 1 à 4 du livre.

S2	**Introduction à l'approche de développement et aux outils à utiliser** Distinction entre le modèle procédural et le modèle objet. Manière d'aborder la solution à développer. Notions fondamentales de la POO. Pile et tas. Instanciation. Usage de classes existantes. Développement de classes simples. **Arrimage aux objectifs**: cette séance touche aux points {1} à {5} des objectifs spécifiques. **Formule pédagogique :** cette séance aura lieu en partie en classe et en partie en laboratoire. **Référence(s)**: **[P1]** chapitres 1 à 4
S3, S4	**Notions fondamentales des objets** Introduction du concept et de la terminologie objet utilisée. Interface publique d'une classe. Élaboration d'un objet. Attributs et méthodes. Encapsulation. Accesseurs et mutateurs. Propriétés. Qualificatifs d'accès. Classe et instances. Membre de classe et membre d'instance. Constantes. Instances immuables. Application à des problèmes simples. **Arrimage aux objectifs**: cette séance touche aux points {2} à {5} des objectifs spécifiques du cours. **Formule pédagogique :** une partie de certaines séances aura lieu en laboratoire, le reste en classe théorique. **Référence(s): [P1]** chapitre 5
S5	**Construction et destruction** Examiner et appliquer la mécanique de construction sous ses différentes déclinaisons, dans un contexte sans héritage. Comprendre les implications de la construction dans le fonctionnement d'un programme. Construction explicite et implicite. Règles de substitution des constructeurs automatiques. Introduction au constructeur de copie. Destruction des instances dans un contexte managé. **Arrimage aux objectifs**: cette séance touche aux points {2} à {5} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 5
S6	**Composition et agrégation; fichiers** Examiner l'organisation d'une relation d'agrégation ou de composition. Distinguer la composition de l'agrégation. Construction du composé et des composants. Application. Usage des classes disponibles dans le système pour interagir avec les fichiers de texte. Lecture et écriture dans un fichier. **Arrimage aux objectifs**: cette séance touche aux points {2} à {5} des objectifs spécifiques du cours. **Référence(s): [P2]** chapitre 20

S7	**Programmation générique; collections** Généralisation de l'idée de type. Exploitation des possibilités de la collection List<>. **Arrimage aux objectifs**: cette séance rejoint touche aux points {5} à {8} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitres 11 et 14 **Gestion des exceptions** Notion d'exception. Catégories d'exception. Hiérarchie disponible dans l'environnement de développement. Lever une exception. Traiter une exception. Relancer une exception. **Arrimage aux objectifs**: cette séance touche aux points {2} à {5} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 10
S8	**Héritage d'implantation** Comprendre la raison d'être de l'héritage d'implantation. Examiner les relations entre parent et enfant et, de manière plus large, entre un ancêtre et sa descendance. Distinguer l'héritage de la composition. Surcharge de méthodes. Emploi explicite des membres d'un parent. Qualificatif d'accès *protected*. Emploi de quelques éléments tirés de la notation UML pour représenter une classe. **Arrimage aux objectifs**: cette séance touche surtout les points {5} et {6} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 6
S9	**Polymorphisme et abstraction** Examiner la notion de polymorphisme et voir comment procéder à son implantation dans une hiérarchie de classes. Saisir l'impact du polymorphisme sur notre manière de développer et de voir les systèmes complexes. Établir les nuances entre polymorphisme et classe abstraite. **Arrimage aux objectifs**: cette séance touche surtout les points {5} à {7} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 6

S10	**Test**
S11	**Héritage d'interface** Différence entre classe abstraite et interface. Interfaces fréquemment utilisées du système. Déclaration d'une interface. Définition des méthodes imposées par l'interface. Règles d'héritage multiple dans le contexte de l'outil utilisé. Parallèles et différences avec d'autres langages POO. **Arrimage aux objectifs**: cette séance touche aux points {5} à {7} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 7
S12	**Délégués et expressions Lambda** Présentation du concept de délégué. Présentation des expressions Lambda et leur utilisation en conjonction avec les collections. Bref aperçu de LINQ. **Arrimage aux objectifs**: ce chapitre touche aux points {4} à {8} des objectifs spécifiques du cours. **Référence(s): [P1]** chapitre 12
S13	**Surcharge des opérateurs et conversions de type** Examiner les caractéristiques de la surcharge d'opérateurs et appliquer les règles propres à la surcharge avec le langage utilisé. Implications de la surcharge des opérateurs. Conversion de type – implicite et explicite. Importance relative de la surcharge dans le langage utilisé. **Arrimage aux objectifs**: cette séance touche aux points {2} à {5} des objectifs spécifiques du cours. **Référence(s): [P2]** chapitre 11
S14	**Modèles de conception courants** Présentation du concept. Avantages d'utiliser des modèles courants dans le développement à petite et à grande échelle. Examen de quelques modèles courants : Singleton, Clonage, Fabrique et Observateur. Si le temps le permet, d'autres modèles de conception seront présentés. **Arrimage aux objectifs**: ce chapitre rejoint les points {5} à {9} des objectifs spécifiques du cours. **Référence(s): [P4]** en adaptant les idées à l'outil du cours
S15	**Examen final**

6. *Évaluation des apprentissages*

Les tests et le contrôle final sont des **évaluations individuelles** et présumeront que chaque membre d'une équipe a contribué activement à la réalisation de l'ensemble de chaque travail pratique et a bien compris les notions reliées à ces travaux.

Description	Pondération
Tests Il y aura deux tests durant la session. Le premier aura lieu vers la séance 5 et comptera pour **15** points. Le deuxième aura lieu à la séance 10 et comptera pour **25** points. **Vous aurez droit à la documentation lors de ces deux tests**. **Les tests sont des évaluations individuelles**.	**40%**
Travaux pratiques et laboratoires Deux travaux pratiques seront à réaliser au cours de la session. Chacun vous demandera, à partir d'un problème soumis, d'appliquer le modèle objet à l'élaboration et à la réalisation de sa solution. Il s'agit de travaux axés sur la programmation objet. Le premier travail sera à réaliser par équipe de deux (2) personnes, et vaudra **10%** de la note finale. Le second travail sera à réaliser par équipe de deux (3) à cinq (5) personnes, et vaudra **20%** de la note finale. Les modalités de ce travail seront précisées au moment où le travail sera distribué aux étudiants.	**30%**
Examen final Un examen final récapitulatif valant **30%** de la note finale aura lieu lors de la dernière séance de la session. Cet examen portera sur l'ensemble de la matière de la session. **Vous aurez droit à la documentation lors de cet examen**. **L'examen final est une évaluation individuelle.**	**30%**

Quelques règles s'appliquent à l'évaluation :

- Toute modification reliée à une date de remise doit avoir été acceptée par le groupe **et** la direction du CeFTI dans un délai de plus d'une semaine avant l'échéance de la remise tel que prévu par les politiques en vigueur.

- **Aucun retard ne sera toléré** dans la remise des travaux pratiques. Tout travail devra être produit dans un français jugé de bonne qualité. Une pénalité allant jusqu'à 10% des points pourra être appliquée à un travail produit dans un français ne rencontrant pas les standards de qualité de la Faculté des sciences. Les règles de

qualité des programmes qui seront mentionnées en cours de session seront applicables aux travaux pratiques et au code rédigé dans le cadre des contrôles.

- L'absence à un test donne droit (!) à la note 0. Il n'y a pas d'évaluation de reprise mais des motifs **sérieux** pourront être pris en considération et un arrangement pourra être proposé dans de telles circonstances.

7. Plagiat

Conformément à l'article 9 du Règlement des études de l'Université de Sherbrooke, le plagiat, soit le fait dans une activité évaluée de faire passer pour sien les idées et le travail d'autrui, est un délit académique qui peut être sanctionné par les autorités disciplinaires compétentes. Peuvent être imposées à titre de sanctions, l'une ou plusieurs des mesures suivantes :

a) la réprimande simple ou sévère consignée au dossier étudiant pour la période fixée par l'autorité disciplinaire ou, à défaut, définitivement. En cas de réprimande fixée pour une période déterminée, la décision rendue demeure au dossier de la personne aux seuls fins d'attester de l'existence du délit en cas de récidive;
b) l'obligation de reprendre une production ou une activité pédagogique, dont la note pourra être établie en tenant compte du délit survenu antérieurement;
c) la diminution de la note ou l'attribution de la note E ou 0;
d) le renvoi du dossier à la personne responsable de l'évaluation d'une production ou d'une activité pédagogique pour qu'elle attribue une nouvelle note en tenant compte du délit.

Par plagiat, on entend notamment :

- copier intégralement une phrase ou un passage d'un livre, d'un article de journal ou de revue, d'une page Web ou de tout autre document en omettant d'en mentionner la source ou de le mettre entre guillemets;
- reproduire des présentations, des dessins, des photographies, des graphiques, des données sans en préciser la provenance et, dans certains cas, sans en avoir obtenu la permission de reproduire;
- utiliser, en tout ou en partie, du matériel sonore, graphique ou visuel, des pages Internet, du code de programme informatique ou des éléments de logiciel, des données ou résultats d'expérimentation ou toute autre information en provenance d'autrui en le faisant passer pour sien ou sans en citer les sources;
- résumer ou paraphraser l'idée d'un auteur sans en indiquer la source;
- traduire en partie ou en totalité un texte en omettant d'en mentionner la source ou de le mettre entre guillemets;
- utiliser le travail d'un autre et le présenter comme sien (et ce, même si cette personne a donné son accord);
- acheter un travail sur le Web ou ailleurs et le faire passer pour sien;
- utiliser sans autorisation le même travail pour deux activités différentes (autoplagiat).

8. Adresse électronique pour les remises de travaux

Mon adresse officielle de courriel de l'Université est **pierre.prudhomme@usherbrooke.ca**. Toutefois, je recommande pour me rejoindre plus efficacement d'utiliser mon adresse personnelle qui est le

p.prudhomme@gmail.com

Vous aurez normalement une réponse beaucoup plus rapidement en procédant par cette voie.

9. Bibliographie et médiagraphie

Ce qui suit se veut une référence assez complète des manuels et documents électroniques à se procurer ou à consulter pour le cours.

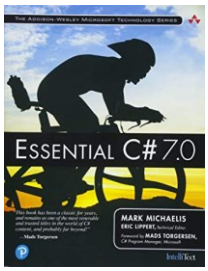
Supports dédiés au cours

Le site Moodle du cours vous donnera accès aux documents écrits par l'enseignant, aux exemples vus en classe sous forme électronique et aux exercices qui seront présentés lors des séances de travaux pratique.

Au surcroît, vous avez accès au site Web suivant qui est maintenu au prix d'efforts remarquables par Patrice Roy que je félicite et remercie au passage. On y trouve des articles de fond fouillés et très pertinents sur le sujet du cours qui vont souvent plus loin encore que ce que nous avons le temps de faire et également d'autres sujets d'intérêt pour des futurs chargés de projets. Ce site se trouve à l'adresse suivante:

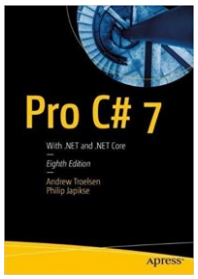
```
http://h-deb.clg.qc.ca/UdeS/
```

Références papier



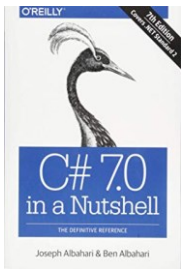
[P1] MICHAELIS, Mark with LIPPERT, Eric; *Essential C# 7.0 (6th edition)*, Addison-Wesley, 2018.
ISBN-10: 1509303588
ISBN-13: 978-1509303588.

Livre suggéré du cours, qui couvre beaucoup plus de notions que ce que nous aurons le temps de voir et dont le niveau est accessible à des étudiants débutants. Toutefois, il ne couvre pas certains aspects que nous aborderons dans le cours.



[P2] TROELSEN, Andrew and JAPIKSE, Phillip; *Pro C# 7 8th Edition*, Apress, 2017.
ISBN-10: 1484230175
ISBN-13: 978-1484230176.

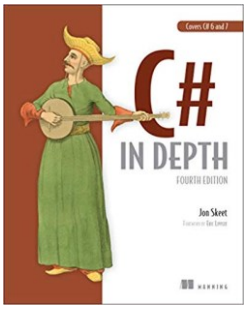
Ce livre couvre d'autres aspects qui ne sont pas abordés par [P1] et lui est donc complémentaire. Intéressant pour celui qui veut approfondir les notions du cours et aller vraiment plus loin. Avoir des connaissances de programmation avant d'aborder ce livre est préférable mais il n'est pas nécessaire d'être déjà un expert pour comprendre les explications fournies.



[P3] ALBAHARI, Joseph & ALBAHARI, Ben; *C# In a Nutshell, The Definitive Reference*, O'Reilly Media, 2017.
ISBN-10: 9352136640
ISBN-13: 978-1491987650.

Ce livre va droit au but en omettant trop souvent les explications qui rendraient le tout compréhensible à un débutant. L'organisation des concepts est perfectible. Une version de poche en guise de référence est également disponible et peut être pratique lorsqu'un rappel d'une notion connue est requis.

[P4] SKEET, Jon; *C# In Depth 4th Edition,* Manning Publications Co., 2019.
ISBN-10: 1617294535
ISBN-13: 978- 1617294532.



Un aperçu des évolutions de C# au fil du temps. Plusieurs approfondissements sur des sujets intéressants mais pas nécessairement à la portée d'un étudiant débutant au DTI. Demande une perspective de l'informatique et de la programmation pour en tirer tout le sens. Pour les intéressé(e)s.

[P5] FREEMAN, Eric & FREEMAN Elisageth & AL.; *Head First Design Patterns: A Brain-Friendly Guide ,* O'Reilly Media, 2004.
ISBN-13: 978- 0596007126.

Un livre très intéressant sur divers modèles de conception courants. A été écrit pour Java ce qui suppose une adaptation pour C# mais comme la philosophie des deux langages est proche, ce n'est rien de majeur. Va quand même plus loin que les objectifs fondamentaux du cours. *Pour les intéressé(e)s*.