Université de Sherbrooke

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

Cours :	INF 737 – Conception orientée objets avancée
Trimestre :	Automne 2023
Enseignant :	Patrice Roy

1. Mise en contexte

Ce cours présume qu'*a priori*, l'étudiant(e) aura un bagage de connaissances sur :

- La programmation traditionnelle procédurale : **découpage**, **procédures** et **fonctions**, les diverses déclinaisons du **passage de paramètres**, etc.
- Les idées fondamentales de la programmation orientée objet (POO) que sont la **classe** et **l'instance**, les **attributs**, les **méthodes**, *etc*.
- Les idées de base associées à la **gestion de la mémoire allouée dynamiquement** et à la **gestion des ressources** de manière générale.
- Certaines des **structures de données** parmi les plus connues, dont le **tableau**, la **liste chaînée**, la **pile** et la **file**.

L'étudiant(e) devrait aussi avoir une connaissance suffisante de la syntaxe de C++ afin de pouvoir comprendre les exemples écrits dans ce langage qui seront présentés durant les cours et effectuer les travaux demandés. Notez que C++, comme Java, C#, JavaScript et plusieurs autres langages, tire en partie sa syntaxe de celle du langage C.

L'approche OO étant le modèle dominant à plusieurs égards dans l'industrie du développement logiciel, et plus spécifiquement dans le développement de logiciels de jeux vidéo, ce cours visera à assurer l'acquisition par l'étudiant(e) d'un bagage préalable à l'ensemble des cours du programme impliquant ce type de développement logiciel spécialisé. D'autres approches (programmation fonctionnelle, programmation générique) s'ajouteront à l'approche OO dans ce cours pour nous permettre de réaliser du code à la fois plus élégant et plus efficace.

2. Place du cours dans le programme

Le modèle objet est un incontournable de presque tout développement informatique en industrie aujourd'hui. La pensée OO telle que nous l'appliquons en informatique sauve du temps, de l'argent, et a amélioré de manière significative la qualité des outils logiciels.

Comme pour la plupart des modes de pensée, bien comprendre le modèle objet implique l'avoir appliqué au préalable pour en saisir les nuances et les enjeux. Se limiter à une couverture abstraite ne suffit pas à développer une compétence opérationnelle. C'est pourquoi nous allons apprendre ici comment programmer selon le modèle OO, avec le langage le plus largement répandu dans le domaine des jeux vidéo.

Le cours s'intégrera conceptuellement aux cours de la même session à titre de fournisseur de stratégies de programmation, et servira aussi de fondation au cours ***INF709 – Concepts spécialisés de programmation en jeu vidéo*** de la session suivante.

3. Objectifs généraux

Ce cours se propose de donner à l'étudiant(e) une *connaissance appliquée* de la programmation et de la philosophie OO, ainsi que de certains éléments de conception logicielle avancée.

Le terme « connaissance appliquée » signifiera à la fois le *savoir* permettant de comprendre les systèmes développés selon les principes sous-jacents au modèle OO qui seront couverts, et le *savoir-faire* requis pour en tirer soi-même profit dans un contexte de développement.

Exprimé en termes généraux, nous présumerons de l'étudiante ou de l'étudiant une connaissance des bases de la POO, et nous viserons à raffiner l'application de ces connaissances face à des problèmes concrets et à l'aide de stratégies éprouvées ou novatrices.

Afin d'atteindre cet objectif, nous développerons une compréhension opérationnelle du langage de programmation C++, qui est le plus répandu dans l'industrie du jeu. Ceci sera un atout pour vous dans votre carrière.

4. Objectifs spécifiques

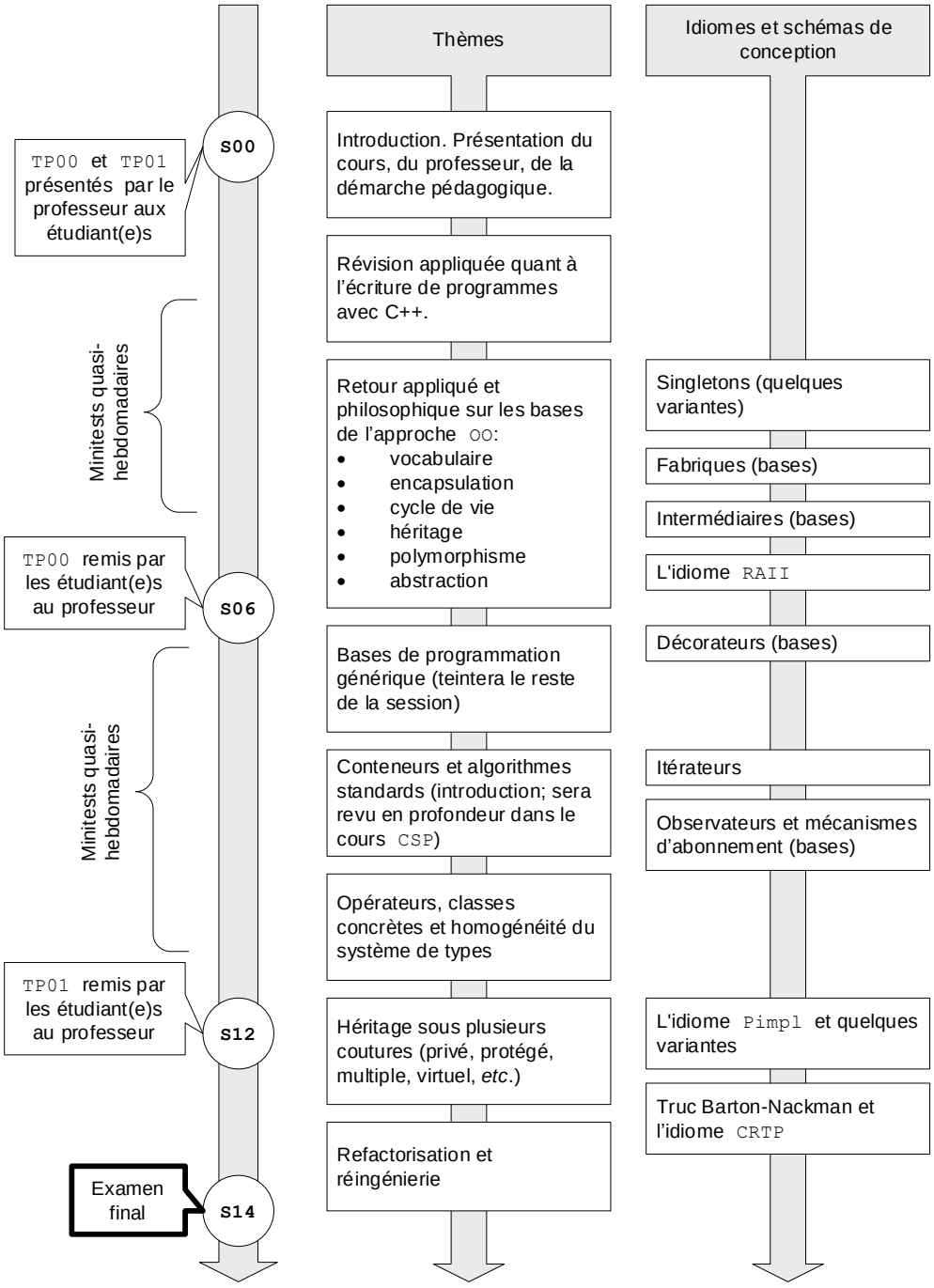
Au terme du cours, l'étudiante ou l'étudiant sera capable :

- D'identifier quel type de problème se prête le mieux à une solution ○○.
- D'expliquer les avantages d'une solution ○○.
- D'analyser un problème pour y offrir une solution ○○.
- De rédiger un programme ○○ afin de résoudre un problème concret.
- De respecter des standards de programmation.
- De tirer profit des divers principes de la programmation ○○.
- De développer à l'aide d'une hiérarchie de classes existante.
- De produire une hiérarchie de classes.
- De généraliser les comportements par polymorphisme.
- De reconnaître les limites de l'héritage et d'en comprendre les alternatives.
- D'utiliser des conteneurs standards.
- D'utiliser des algorithmes.
- D'appliquer un schéma de conception (*Design Pattern*) et, de manière générale, le modèle ○○ à un problème de développement logiciel.
- De reconnaître une opportunité d'application judicieuse d'un schéma de conception.

5. Planification hebdomadaire

Le cours suivra une philosophie appliquée. Nous commencerons par un volet révision mêlant concepts et exercices, puis nous plongerons dans des stratégies pour appliquer efficacement l'approche ○○ et pour montrer qu'il est possible d'écrire du code à la fois propre et efficace. C'est dans cette optique d'application efficace que nous examinerons quelques schémas de conception et quelques idiomes de C++.

La structure prévue pour la session suivra le schéma ci-dessous, avec ajustements au besoin (et il y en aura). Ce schéma se veut plus illustratif que directif, et nous prendrons des libertés en fonction du rythme de la classe. Notez que les *minitests* apparaîtront sans prévenir et ne sont donc pas identifiés dans le schéma.



6. Approche pédagogique préconisée

Pour favoriser l'intégration des nombreux concepts au menu, nous suivrons essentiellement le modèle suivant :

- Exposés magistraux en classe (possibilité de classe « virtuelle » en fonction des conditions sanitaires), où les étudiant(e)s sont *fortement* encouragé(e)s à contribuer par leurs questions et commentaires.
- Travaux pratiques et exercices à teneur formative, qui permettront aux étudiant(e)s de mesurer concrètement leur compréhension de la matière explorée, et qui pourront être corrigés par le professeur ou encore *autocorrigés* à l'aide d'une grille de vérification, selon le cas.
- Travaux pratiques à teneur sommative, évalués par le professeur en fonction des mêmes critères que ceux appliqués dans le cadre des activités formatives;
- Des questions de réflexion (et parfois à saveur technique) sur une base quasi hebdomadaire, et
- Un contrôle théorique récapitulatif, en toute fin de parcours, vérifiant formellement l'atteinte des objectifs.

Les questions et contrôles seront conçus en tenant compte du fait que chaque membre d'une équipe aura contribué activement à la réalisation de chacun des travaux pratiques et aura bien compris les implications philosophiques et techniques de ces travaux.

7. Évaluation de l'apprentissage

Les évaluations sommatives seront réparties et pondérées comme suit.

Minitests ***(***40%***)***

Au moins huit ***minitests***, souvent d'une seule question, auront lieu sur une base relativement régulière, soit une par semaine, la plupart des semaines.

Chaque question portera sur le thème de la semaine précédente ou des deux semaines précédentes, Les questions seront surtout orientées sur la réflexion, mais la technique à proprement dit s'y glissera à l'occasion.

Le poids de chacun de ces petits tests sera 5% de la note finale. En général, le temps alloué pour y répondre sera d'environ 15 minutes, au début du cours. Si plus de huit minitests ont lieu, alors les huit meilleurs résultats seront conservés pour chaque étudiant(e).

Travaux pratiques ***(***30%***)***

Deux **travaux pratiques** seront à réaliser au cours de la session. Chacun vous demandera, à partir d'un problème concret, d'appliquer le modèle objet à l'élaboration et à la réalisation de sa solution. Chacun demandera une part importante de programmation.

En pratique, nous appliquerons les techniques et concepts du cours au petit projet de jeu vidéo réalisé dans les autres cours de la session, ou encore au cours d'intelligence artificielle (à votre choix). Ce que vous livrerez sera chaque fois un rapport individuel démontrant, code à l'appui, votre capacité de réaliser un transfert de connaissances et d'appliquer, de manière pertinente et imaginative, la matière du cours INF737.

Chaque travail aura un poids de 15% sur la note finale.

Examen ***(***30%***)***

Un **examen final** récapitulatif valant 30% de la note finale aura lieu lors de la dernière séance de la session.

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 5% 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 distribuées en cours de session seront applicables aux travaux pratiques et au code rédigé dans le cadre des contrôles.

Un pourcentage de chaque extrant sera consacré à la qualité du français et au format (présentation).

Toute modification reliée à une date de remise doit avoir été acceptée par le groupe et la direction du CeFTI dans un délai plus grand qu'une semaine avant l'échéance de la remise.

8. Plagiat

Un document dont le texte et la structure se rapportent à des textes intégraux tirés d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction de tout travail individuel ou de groupe une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme « le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de l'œuvre d'autrui. » Le cas échéant, le plagiat est un délit qui contrevient à l'article 8.1.2 du Règlement des études : « tout acte ou manœuvre visant à tromper quant au rendement scolaire ou quant à la réussite d'une exigence relative à une activité pédagogique. ». À titre de sanction disciplinaire, les mesures suivantes peuvent être imposées : a) l'obligation de reprendre un travail, un examen ou une activité pédagogique, et b) l'attribution de la note E ou de la note 0 pour un travail, un examen ou une activité évaluée. Tout travail suspecté de plagiat sera référé à la vice-doyenne à l'enseignement de la Faculté des sciences.

9. Adresse électronique pour remise des travaux

Patrice.Roy@USherbrooke.ca

10.Médiagraphie

Les notes de cours qui vous seront distribuées constitueront votre référence principale pour la session qui s'annonce. D'autres références vous sont proposées sous la forme d'une *médiagraphie* commentée sur le site suivant :

> http://h-deb.clg.qc.ca/Liens/Suggestions-lecture.html

Parmi les quelques suggestions que vous trouverez sur ce site, portez particulièrement attention aux volumes suivants :

- **Effective C++**, **Effective Modern C++** et **Effective `STL`**, de ***Scott Meyers***
- **Exceptional C++**, de ***Herb Sutter***
- **Modern C++ Design**, par ***Andrei Alexandrescu***
- **C++ Coding Standards**, par ***Herb Sutter*** et ***Andrei Alexandrescu***
- **The C++ Programming Language**, par ***Bjarne Stroustrup***
- **The C++ Standard Library – A Tutorial and Reference, 2nd Edition**, par ***Nicolai M. Josuttis***
- **C++ 17 The Complete Guide**, par ***Nicolai M. Josuttis***
- **C++ Software Design: Design Principles and Patterns for High-Quality Software**, par ***Klaus Iglberger***

Certaines de ces références ne couvrent pas les révisions de C++ depuis C++ 03, mais sachez que dans ce cours (et dans les notes de cours), nous utiliserons C++ 20, avec quelques incursions en C++ 23. Vous ne serez donc pas en reste.

Site Web du cours et autres références

Le site Web du cours devrait être, avec les notes de cours en tant que telles, votre référence principale. Vous y trouverez *beaucoup* de matériel :

> http://h-deb.clg.qc.ca/UdeS/COA/

Je déposerai entre autres sur ce site des références électroniques me semblant pertinentes. Je vous invite donc fortement à le consulter sur une base régulière.

Mon site principal est :

> http://h-deb.clg.qc.ca/

Dans tous les cas, faites attention aux majuscules et aux minuscules puisque l'adresse est sensible à la casse.