Université de Sherbrooke

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

CeFTI
INF737

Accompagnement au plan de cours

Note : ce document accompagne le plan de cours et vise à apporter de l'information contextuelle sur le cours. L'information d'ordre administratif est disponible dans le plan de cours en soi.

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**, les idées de **cohésion** et de **couplage**, *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++ pour 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, alors une familiarité avec l'un ou l'autre de ces langages devrait suffire à démarrer les travaux.

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, plus efficace et profitant mieux du matériel contemporain que ne le permettrait la POO à elle seule.

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 économise du temps, de l'argent, et a contribué à améliorer 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. Le cours utilisera C++ dans son acception contemporaine et profitera des plus récents raffinements apportés à ce langage lorsque cela sera opportun.

4. Objectifs spécifiques

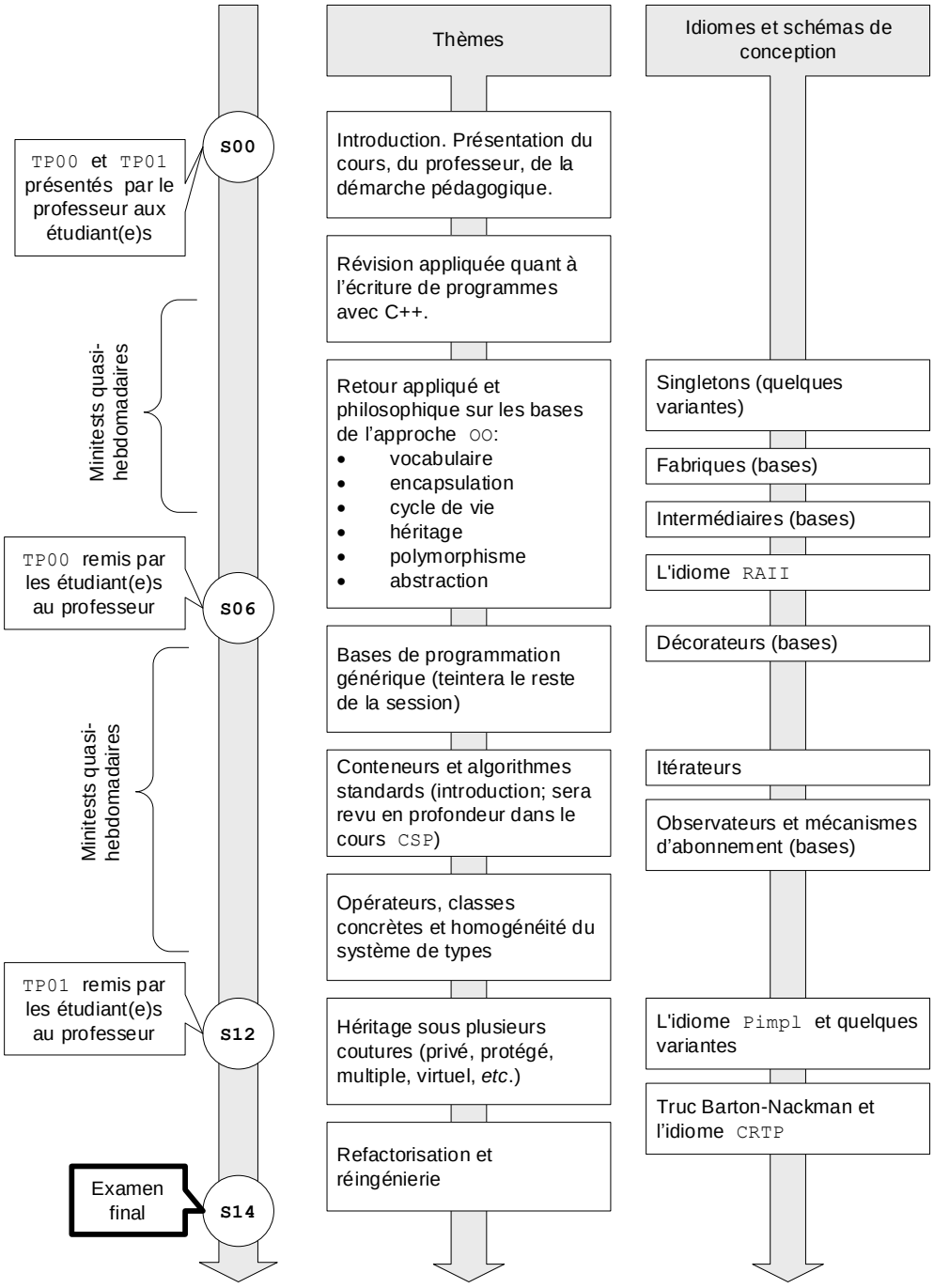
Au terme du cours, l'étudiante ou l'étudiant sera capable :

- D'identifier quel type de problème se prête le mieux à une solution OO.
- D'expliquer les avantages d'une solution OO.
- D'analyser un problème pour y offrir une solution OO.
- De rédiger un programme OO afin de résoudre un problème concret.
- De respecter des standards de programmation.
- De tirer profit des divers principes de la programmation OO.
- 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'exprimer des solutions sur la base d'algorithmes.
- D'appliquer un schéma de conception (*Design Pattern*) et, de manière générale, le modèle OO à un problème de développement logiciel.
- De reconnaître une opportunité d'application judicieuse d'un schéma de conception.

5. Planification générale

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 OO 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. Adresse électronique pour remise des travaux

Patrice.Roy@USherbrooke.ca

7. Médiagraphie

Le site Web du cours constituera votre référence principale pour la session qui s'annonce. De plus, vous voudrez sans doute porter 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 (peut-être même des bribes de C++ 26 si vous êtes gentilles et gentils!) 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.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.ca/

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