UNIVERSITÉ DE SHERBROOKE
Département d'informatique

IFT 232
Méthodes de conception orientées objet

Plan de cours
Automne 2016

Enseignant

Mikaël Fortin
Courriel : Mikael.Fortin@USherbrooke.ca
Local : D4-2004
Téléphone : (819) 821-8000 poste 66106
Site : Lecteur réseau public
Disponibilité : à déterminer

Horaire

Exposé magistral :	Mardi 13 h 30 à 15 h 20	salle D3-2038
	Mercredi 10 h 30 à 12 h 20	salle D3-2038

Description officielle de l'activité pédagogique[1]

Objectifs — Spécifier, concevoir et tester des composants logiciels. Tester l'intégration des composants. Mesurer la qualité de la conception. Appliquer le processus de conception au sein d'un cadre documenté et normalisé.

Contenu — Types abstraits algébriques. Critères de conception. Encapsulation, héritage et polymorphisme. Critères de composition en classes, schémas de conception (*design patterns*) et cadres d'application (*frameworks*). Documentation de la conception avec la notation UML. Techniques de revue de conception. Tests unitaires. Programmation par composants. Mesure de qualité de la conception. Introduction à la programmation agile.

Crédits 3
Organisation 3 heures d'exposé magistral par semaine
1 heure d'exercices par semaine
5 heures de travail personnel par semaine
Préalable GIF600 ou IFT159

[1] http://www.usherbrooke.ca/fiches-cours/ift232

1 Présentation

Cette section présente les objectifs et le contenu détaillé du cours. Cette section représente la description officielle du cours telle qu'adoptée par les comités de programme du département d'informatique.

1.1 Mise en contexte

La programmation orientée objet (POO) est un paradigme informatique considéré comme une des innovations les plus importantes en génie logiciel[2]. À travers SIMULA, les origines de la POO remontent aussi loin qu'en 1967. La POO a donné par la suite naissance à d'autres innovations significatives en informatique comme le modèle MVC (1980), les schémas de conception (1991) et la refactorisation (1993). La POO repose sur 5 concepts fortement intégrés qu'elle articule en un tout cohérent : objet (encapsulation), classe (instanciation et réutilisation), message (communication), héritage (réutilisation) et méthode (comportement). Comme tout paradigme, la POO a développé sa propre culture, sa propre façon de concevoir le monde : on n'écrit pas un programme objet comme on écrit un programme fonctionnel. Cette culture prend toute sa mesure dans les méthodologies de développement agile.

1.2 Objectifs spécifiques

L'objectif du cours IFT232 est de développer un point de vue intégré et vertical de la culture de la conception par objets : de la conception jusqu'au programme final, à sa maintenance et à son évolution. A la fin du cours, un étudiant ou une étudiante sera capable, en suivant une méthodologie agile, d'analyser en termes de POO un problème de taille moyenne, de concevoir une solution adaptée, de la programmer en objets et de la faire évoluer. Il ou elle devra être en outre capable de la valider et d'en juger la qualité à l'aide de critères bien définis. Il est à noter que le but du cours n'est pas d'apprendre un langage, mais bien le processus menant à résoudre un problème en utilisant les concepts fondamentaux de la conception et de la programmation par objets.

À la fin de cette activité pédagogique, l'étudiante ou l'étudiant sera capable :

1. Maîtriser un langage de spécification (UML) ;
2. Maîtriser les concepts de base de la programmation par objets ;
3. Maîtriser le processus de refactorisation d'un programme ;
4. Maîtriser la notion de schémas de conception ;
5. Maîtriser les processus de tests ;
6. Connaître diverses approches de la conception orientée objet ;
7. Comprendre et appliquer les concepts principaux d'une méthode de développement agile ;
8. Être capable d'évaluer la qualité de conception d'un programme.

[2] David A. Wheeler, « The Most Important Software Innovations », révision du 22 juillet 2011, http://www.dwheeler.com/innovation/innovation.html

1.3 Contenu détaillé

Thème	Contenu	Heures	Objectifs
1	**Programmation par objets** • Les principes de base de la programmation par objets : instanciation, encapsulation, héritage, polymorphisme, transmission de messages… • Application aux langages de programmation	8	2
2	**Spécifications** • Types abstraits algébriques • Programmation par contrats • Spécification de programmes; préconditions, postconditions, invariants…	3	5, 6
3	**Tests**: • Test unitaires • Application aux langages de programmation	3	5
4	**Méthodes de conception** • Méthodes de conception agiles	3	6, 7
5	**Documentation de la conception** • UML	3	1
6	**Refactorisation** • Critères de conception et qualité de la conception • Techniques de refactorisation	3	3, 4, 8
7	**Programmation par composants** • Exemple d'application	4	4, 6
8	**Schéma de conception** • Principes généraux • State, Singleton, Observer, Adapter, Command, Composite, Abstract Factory, Factory Method, Template Method…	12	4, 8
9	**Cadre de travail** • Principes et applications • Étude de cas	3	4, 6, 8
10	**Projet de session** • Remise, présentation, démonstration	6	1-8

2 Organisation

Cette section présente la méthode pédagogique utilisée, le calendrier officiel du cours, la méthode d'évaluation ainsi que l'échéancier des travaux.

2.1 Méthode pédagogique

Une semaine comprend 4 heures de présence en classe. Sur toute la session, il y aura en moyenne trois heures de cours dit théorique et une heure d'exercices. Des séances d'exercices de 2 heures alterneront avec les blocs de cours théoriques. Lorsque le projet sera commencé, les heures d'exercices seront consacrées au projet pour permettre aux équipes d'organiser leur travail et de se rencontrer plus facilement. Le cours a une approche pratique et met l'accent sur l'implémentation des solutions conçues. Tout le matériel pédagogique sera placé sur Moodle.

2.2 Calendrier du cours

	Semaine du	**Thème**	**Travaux**	**Laboratoires**
1	29-08-2016	1	Début TP1	Labo 0 : Introduction à Java
2	05-08-2016	1		
3	12-09-2016	2,3		Labo 1 : Programmation orientée objet
4	19-09-2016	3		
5	26-09-2016	3,6	Début de la préparation des projets Remise du TP1, Début TP2	Labo 2 : Tests unitaires
6	03-10-2016	6		Labo 3 : Réusinage
7	Période du 08-10-2016 au 15-10-2016	Examen Intra	Remise TP2 (Vendredi avant les examens)	
8	17-10-2016	Relâche		
9	24-10-2016	4	Limite d'approbation des projets (avant la fin de la semaine). Début de production des projets.	
10	31-10-2016	5,8	Début TP3	Labo 4 : Patron Observer
11	07-11-2016	8		
12	14-11-2016	8	Remise TP3, Début TP4	Labo5 : Templates et Factories
13	21-11-2016	8	LIMITE : Début production des projets	
14	28-11-2016	7,9	Remise TP4, suite projet	Labo 6 : Patron Command
15	05-12-2016	10	Projet	
17	Période du 13-12-2016 au 23-12-2016	Examen final	Remise des projets	

2.3 Évaluation

Devoirs (4) : 20 %
Projet : 20 %
Examen intra : 25 %
Examen final: 35 %

2.3.1 Échéancier des travaux

TP	Thème	Réception du TP	Remise du TP
1	Concepts P.O.O.	29-08-2016	26-09-2016
2	Tests et refactorisation	26-09-2016	08-10-2016
3	Patrons de conception #1	31-10-2016	14-11-2016
4	Patrons de conception #2	14-11-2016	28-11-2016
Projet	Préparation et approbation	26-09-2016	29-10-2016
Projet	Production	24-10-2016	13-12-2016

2.3.2 Qualité du français et de la présentation

Conformément à l'article 17 du règlement facultaire d'évaluation des apprentissages[3], l'enseignant peut retourner à l'étudiante ou à l'étudiant tout travail non conforme aux exigences quant à la qualité de la langue et aux normes de présentation.

2.3.3 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[4] : « 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 au Secrétaire de la Faculté des sciences.

2.4 Directives particulières pour les travaux pratiques

Les travaux pratiques sont effectués seul(e) ou par équipe de deux étudiant(e)s. Les travaux pratiques sont réalisés en Java. L'environnement de programmation intégré Éclipse (http://www.eclipse.org/) est disponible au département d'informatique. L'environnement de tests utilisé est JUnit (http://www.junit.org/). Les travaux pratiques peuvent être réalisés sous tout autre environnement, mais le code et les tests devront fonctionner sous Eclipse et Java SE 6 et JUnit 4 au moment de la remise du travail. Les énoncés des travaux seront rendus disponibles sur le lecteur réseau public. La remise du travail (code, tests et autres documents) sera effectuée par turnin. La remise du travail s'effectue au jour et à l'heure exigés.

2.5 Directives particulières pour le projet

Le projet se fait par équipe de 6 à 10 personnes. Une équipe doit idéalement comporter un nombre pair de membres. Chaque équipe définit son propre projet. Les projets peuvent être réalisés dans le langage à objets et l'environnement de développement de votre choix. À la mi-session, le professeur doit approuver la composition de l'équipe, la définition du projet et le langage utilisé. Pour la remise du projet sont demandés une démonstration, le code source et les tests unitaires, un rapport pour le groupe, ainsi qu'un rapport individuel. La remise de tous les documents et fichiers liés au projet se fait par turnin.

3 Matériel nécessaire pour le cours

Les manuels et articles en référence ne sont pas obligatoires. Le langage utilisé dans le cours est Java. Les projets peuvent être réalisés dans le langage à objets le plus approprié à la nature du projet, e.g. existence d'une bibliothèque donnée.

[3] http://www.usherbrooke.ca/sciences/intranet/informations-academiques/reglement-devaluation/

[4] http://www.usherbrooke.ca/programmes/etude

4 Références

Documents recommandés

1. J. Bloch, Effective Java Programming Language Guide, 2e édition, Addison-Wesley, 2008.
2. L. Audibert, *UML 2 : de l'apprentissage à la pratique*, Paris : Ellipses, 2009.
3. J.-L. Bénard, *Gestion de projet eXtreme Programming*, Paris : Eyrolles, 2005.
4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*, Addison-Wesley, 1994.

Manuels et documents complémentaires

1. T. Budd, *An introduction to object-oriented programming* , 3e édition, Addison-Wesley, 2002.
2. H. Bersini, *L'orienté objet*, 2e édition, Paris : Eyrolles, 2004.
3. S. R. Schach, *Object-oriented and classical software engineering*, 8th edition, New York : McGraw-Hill, 2011.
4. C. Larman, *UML 2 et les design patterns*, 3e édition, Pearson Education, 2005.
5. B. Charroux, A. Osmani, Y. Thierry-Mieg, *UML 2 : pratique de la modélisation*, 3e édition, Addison-Wesley, 2010.
6. E. Freeman, E.Freeman, K. et B. Bates, *Design patterns : tête la première*, Paris: O'Reilly, 2005.
7. J. Kerievsky, *Refactoring to patterns*, Boston : Addison-Wesley, 2005.
8. Refactoring to patterns catalog, http://www.industriallogic.com/xp/refactoring/catalog.html.
9. M. Fowler, *Refactoring, improving the design of existing code*, Reading, MA : Addison Wesley, 2000.
10. Refactoring Home Page, http://www.refactoring.com/.
11. K. Beck, *Test driven development : by example*, Boston : Addison-Wesley, 2003.
12. K. Beck, *Extreme programming explained*, Reading, MA : Addison-Wesley, 2000.
13. Extreme programming, A gentle introduction, http://www.extremeprogramming.org/.