UQAC

Département d'informatique
et de mathématique
Université du Québec à Chicoutimi

PLAN DE COURS

8INF957

Programmation objet avancée

Département d'informatique et de mathématique

Objectifs généraux du cours

- Comprendre et maîtriser les fondements de formation de l'approche objet pour évaluer et améliorer la qualité de logiciels.
- Identifier et comprendre les avantages et inconvénients de l'approche objet.
- Concevoir des nouveaux types selon le besoin, en vue de réponde aux exigences d'un système.
- Comprendre la conception et le développement orientés objet, aspect, composant, contrat, ou autres.
- Introduire l'architecture et l'adaptation de logiciels et assurer leur qualité, en particulier: faciliter l'adaptation de logiciels, améliorer la réutilisation de composantes logicielles et faciliter la maintenance.
- Offrir une occasion de *critiquer* des articles existants et de *résoudre* des problèmes identifiés et relatifs au développement orienté objet, aspect, composant au contrat.
- Encourager les étudiants à détecter des défis liés à l'approche objet, à préparer et à présenter des exposés pour montrer ces défis. Les étudiants sont invités à présenter les approches existantes pour résoudre tels défis et à proposer des solutions.
- Offrir une vision globale liée à plusieurs domaines de recherche en informatique liés à l'application de patrons et modèles : invitation des chercheurs ou collègues pour présenter leurs recherches.

Description du cours (objectifs spécifiques)

Ce cours optionnel est destiné aux étudiants ayant déjà suivis un cours de programmation. Il permet d'acquérir les compétences avancées de la programmation par objets OOP comme la programmation orientée aspect AOP, la programmation orienté sujet SOP, la programmation orientée composant COP, la programmation par contrat (OCL), et bien d'autres. Les sujets couverts incluent entre autres

- Rappel sur l'approche objet : classes versus objets, héritage, généricité, polymorphisme, exceptions.
- Étudier la qualité de logiciel relative au paradigme objet (ex. cohésion, couplage, modularité).
- Classe internes, expression Lambda, sérialisation, réflexion, interface Java, interface graphique, multithreading, Méta-programmation, OOP, AOP, SOP, OCL, COP, patrons de conception, ainsi que flux d'entrées/sorties.
- Introduire la programmation orientée composant, l'architecture et l'adaptation logicielle.

À la fin du cours, l'étudiant devrait pouvoir

- collaborer et communiquer avec les membres d'une équipe de développement de logiciel dans la réalisation d'un projet complexe, en se basant sur ces approches.
- améliorer la qualité de logiciels
- être en mesure de mener une démarche de la créativité dans la programmation objet avancée et l'amélioration de qualité de logiciels en tenant compte de contraintes temporelles et budgétaires d'un projet.

Cours magistraux

Il est possible de modifier l'ordre des chapitres dans le calendrier afin de permettre aux étudiants d'avancer dans la partie pratique en parallèle avec la partie théorique du cours. Le contenu de ce plan est préliminaire et pourrait être modifié selon l'avancement durant la session. Le cours permettra aux étudiants l'apprentissage des éléments suivants (voir l'annexe 1 pour plus de détails) :

Chapitre 1. Présentation du cours, plan, projets, et organisation

Chapitre 2. Rappel à l'approche objet, Qualité de logiciel, classes versus objets, héritage, généricité, polymorphisme, exceptions

Chapitre 3. : Approche objet, nio, Fichier, Flux (stream)

Chapitre 4. Classes internes, expression lambda, interface Java

Chapitre 5. Sérialisation, réflexion, réseau avec objet

Chapitre 6. Interface graphique

Chapitre 7. Programmation concurrente, Multithreading

Chapitre 8. Meta-programmation, annotation, JSP

Chapitre 9. Patrons de conception (DP), patrons architecturaux, Introduction à l'architecture des applications

Chapitre 10. Séparation de préoccupations, AOP (AspectJ), SOP (HyperJ), VOP

Chapitre 11. Programmation orientée composant: EJB (Java EE), objet distribué (RMI), Corba

Chapitre 12. Programmation orientée composant OSGi, Spring

Chapitre 13. Programmation par contrat (OCL)

Insertion du cours dans le programme

Le cours est optionnel dans les divers programmes des cycles supérieurs du département d'informatique et de mathématique (DIM) de l'université. Ce cours nécessite la connaissance des concepts de base orientés objets, langages objet (Java, C/C++, C#), Eclipse, etc. De plus, il demande une capacité d'analyse rigoureuse et méthodique.

)

Situation du cours dans le programme

Le cours 8INF957- Programmation objet avancée est un cours optionnel offert à tous les cycles supérieurs dans le département d'informatique et de mathématiques. Aucun cours n'est préalable au cours 8INF957.

Références

1. Notes du cours 8INF957.
2. Doudoux, J.-M. (2014). *Développons avec Java*.
   Accès à la version électronique: http://go.uqac.ca/xPN8
3. Delannoy, C. (2014). *Programmer en Java* (9e éd.). Paris: Eyrolles.
   Accès à la version électronique: http://go.uqac.ca/GJnc
4. Bersini, H. (2013). *La programmation orientée objet : Cours et exercices UML 2 avec Java, C#, C++, Python, PHP et LINQ*. Paris, France: Eyrolles.
   Accès à la version électronique: http://go.uqac.ca/HGpm
5. Bass, L. (2013). *Software architecture in practice* (3e éd.). Boston, MA: Addison-Wesley.
   Disponibilité à la bibliothèque: http://go.uqac.ca/5pit
6. Blanc, X. (2005). *MDA en action : ingénierie logicielle guidée par les modèles*. Paris, France: Eyrolles.
   Accès à la version électronique: http://go.uqac.ca/9hkp
7. Fowler, M. (2006). *Writing Software Patterns*.
   Accès à la version électronique: http://go.uqac.ca/TAXL
8. Gamma, E. (1995). *Design patterns : elements of reusable Object-Oriented software*. Reading, Mass.: Addison-Wesley.
   Disponibilité à la bibliothèque: http://go.uqac.ca/KXeN
9. Gamma, E., & Lasvergères, J.-M. (1999). *Design patterns : catalogue de modèles de conception réutilisables*. Paris: Vuibert.
   Disponibilité à la bibliothèque: http://go.uqac.ca/rmWj
10. Horstmann, C.S. (2006). *Object-oriented design & patterns* (2e éd.). Hoboken, NJ: Wiley.
    Disponibilité à la bibliothèque: http://go.uqac.ca/Atda
11. Kircher, M., & Jain, P. (2004). *Patterns for resource management (Pattern-oriented software architecture, vol. 3)*. Chichester, England: Wiley.
    Disponibilité à la bibliothèque: http://go.uqac.ca/QiqL
12. Metsker, S.J. (2006). *Les Design Patterns en Java : les 23 modèles de conception fondamentaux*. Paris, France: CampusPress.
    Disponibilité à la bibliothèque: http://go.uqac.ca/UtGp
13. Oracle. (n.d.). *Java Platform, Standard Edition 7 API Specification*.
    Accès à la version électronique: http://go.uqac.ca/EvQJ
14. Oracle. (n.d.). *The Java Tutorials*.
    Accès à la version électronique: http://go.uqac.ca/evaV
15. Völter, M., Kircher, M., & Zdun, U. (2005). *Remoting patterns foundations of enterprise, internet and realtime distributed object middleware*. Chichester, England: John Wiley.
    Disponibilité à la bibliothèque: http://go.uqac.ca/CQ5w