COORDONNATEUR		SÉGUIN, Normand	seguin.normand@uqam.ca	(514) 987-3000 4345	PK-4325
GROUPES	10	LAFOREST, Louise	laforest.louise@uqam.ca	(514) 987-3000 7790	PK-4725
		Lundi et mercredi, de 13h30 à 15h00 (cours) – Mercredi, de 15h30 à 17h30 (ateliers)			
	20	DOUKOURE, Ismaël	doukoure.ismael@uqam.ca	(514) 987-3000 3699	PK-4115
		Mardi, de 18h00 à 21h00 (cours) – Mercredi, de 18h00 à 20h00 (ateliers)			

DESCRIPTION

Approfondir les concepts de la programmation orientée-objet. Approfondir les concepts de mise au point et de test de composants logiciels. Identification et définition des classes d'une solution logicielle. Relations entre les classes: composition et héritage. Classes abstraites et polymorphisme. Introduction à la notation UML. Algorithmes récursifs simples. Structures de données classiques: piles, files, listes et arbres binaires de recherche. Techniques classiques de recherche (séquentielle et binaire) et de tri. Introduction à la programmation des interfaces graphiques (GUI). Gestion des événements et des exceptions. Conception de paquetages. Introduction aux outils automatisés de validation.

Ce cours comporte une séance obligatoire de laboratoire (2 heures). Six de ces laboratoires seront évalués.

Préalables INF1120 Programmation I

OBJECTIFS

- Approfondir les méthodes de conception et de programmation orientée-objet
- Spécialisation des classes
- Connaître les structures de données fondamentales et savoir les choisir et les utiliser
- Connaître les techniques de base de la recherche de données et des tris
- Connaître la conception d'une interface graphique (GUI) avec les composants de base
- Acquérir des connaissances de base en génie logiciel
- S'initier aux étapes de la réalisation d'un produit logiciel
  - ❏ Stratégies de conception, de mise au point et de tests
  - ❏ Documentation du logiciel
- Approfondir le langage Java
- À la fin de la session, l'étudiant(e) devrait être en mesure d'élaborer un programme structuré et fonctionnel en utilisant les notions de génie logiciel étudiées au cours et en se servant des différentes structures de données fondamentales.

ÉVALUATION

Description sommaire	Date	Pondération
Examen commun intra		30%
Examen commun final		25%
2 travaux pratiques : TP1 (15%) – TP2 (15%)	Les dates de remise sont spécifiques à chacun des groupes	30%
Quiz en classe	6 quiz (les 5 meilleurs seront retenus)	15%

Les dates de distribution des énoncés des travaux pratiques ainsi que les dates de remise sont spécifiques à chacun des groupes.

Règles concernant le seuil de passage

L'étudiant doit obtenir une moyenne cumulée aux examens égale ou supérieure à 50%, ainsi qu'une moyenne cumulée pour les travaux supérieure ou égale à 50%. Si ces seuils ne sont pas atteints, la mention échec sera automatiquement attribuée au cours.

Travaux pratiques

- ❏ Les règlements de l'UQAM concernant le plagiat seront strictement appliqués. Les travaux pratiques sont strictement individuels.
- ❏ En cas de doute sur l'originalité des travaux, un test oral pourra être exigé. Tous les cas de plagiats seront référés au comité de discipline de la Faculté. La sanction peut aller de la note 0 pour le travail ou pour l'examen jusqu'à l'exclusion de l'université.
- ❏ Une pénalité de 10 % par jour ouvrable sera appliquée aux travaux remis après les dates prévues. Après 5 jours ouvrables de retard, le travail sera considéré comme non remis entraînant la note 0 pour ce travail.
- ❏ Il est de la responsabilité de l'étudiant de faire des copies de sauvegarde de ses travaux sur au moins 2

disquettes. Il est possible que le professeur demande à l'étudiant de lui transmettre certaines parties du travail suite à la remise.

Les étudiants doivent consulter régulièrement le site Web des cours de programmation. On y trouve, entre autres, les énoncés des travaux pratiques, certains exemples et les questions de révision pour les examens.

Nous rappelons aux étudiants qu'ils doivent s'attendre à fournir une moyenne de 6 heures de travail personnel par semaine pour un cours de trois crédits (total de 90 heures).

Politique d'absence aux examens

Un étudiant absent à un examen se verra normalement attribuer la note zéro pour cet examen. Cependant, si l'étudiant était dans l'impossibilité de se présenter à l'examen pour un motif valable, certains arrangements pourront être pris avec son enseignant. Pour ce faire, l'étudiant devra présenter à son enseignant l'un des formulaires prévus à cet effet accompagné des pièces justificatives appropriées (par ex., attestation d'un médecin que l'étudiant était dans l'impossibilité de se présenter à l'examen pour des raisons de santé, lettre de la Cour en cas de participation à un jury).

Une absence pour cause de conflit d'horaires d'examen n'est pas considérée comme un motif valable d'absence, à moins d'entente préalable avec la direction du programme et l'enseignant durant la période d'annulation des inscriptions avec remboursement : tel qu'indiqué dans le guide d'inscription des étudiants, il est de la responsabilité d'un étudiant de ne s'inscrire qu'à des cours qui ne sont pas en conflit d'horaire.

Pour plus de détails sur la politique d'absence aux examens du Département d'informatique et pour obtenir les formulaires appropriés, consultez le site web suivant :
http://www.info.uqam.ca/enseignement/politiques/absence-examen

CONTENU

1. Retour sur le concept de classe
   - Variable d'instance, de classe et valeur constante
   - Accès aux composants d'une classe ·
   - Notion d'encapsulation ·
   - Méthode d'instance et de classe
   - Constructeur
2. Notion d'héritage
   - Contexte d'utilisation
   - Classe Object
   - Constructeur de la sous-classe et Super
   - Héritage et méthodes : redéfinition, surcharge
   - Contrôler l'héritage : Final
   - Notion de polymorphisme
   - Conversion de type entre les classes d'une hiérarchie
3. Javadoc
   - Utilisation des balises courantes
   - Génération de la documentation
4. Introduction aux collections
   - Notion de *collection* en Java
   - Notion de type abstrait (ADT)
   - Notion de structure de donnée

   a) Introduction à ArrayList

   b) Notion d'interface
      - ✓ Notion de contrat
      - ✓ Utilisation vs implémentation

   c) Introduction à Pile (ADT)
      - ✓ Interface des services sur Pile
      - ✓ Exemples d'utilisation

- ✓ Implémentation avec ArrayList

d) Introduction à File (ADT)
- ✓ Interface des services sur Pile
- ✓ Exemples d'utilisation
- ✓ Implémentation avec ArrayList

e) Notion de package

5. Introduction aux listes chaînées
   - Avantages
   - Notion de *Noeud* ou maillon de liste
   - Notion de chaînage
   - Nouvelles implémentations de ADT Pile et File avec liste chaînée
   - Insertion et retrait dans une liste chaînée
   - Notion de remorque de fin de liste
   - Parcours d'une liste
   - Liste ordonnée
   - Notion de liste doublement chaînée
6. Introduction à la récursitivé
   - Fonctionnement : cas de base et convergence
   - Trace d'exécution
   - Conception d'une méthode récursive
   - Conversion d'une méthode récursive en itératif
7. Techniques de recherche
   - Recherche séquentielle non ordonnée et ordonnée
   - Recherche binaire
   - Notion du grand O
   - Arbre binaire de recherche
     - ✓ Insertion
     - ✓ Parcours; prefixe, infixe, postfixe
8. Techniques de tri
   - Tris simples ($O(n^2)$)
     - ✓ Sélection
     - ✓ Échanges (tri à bulles et tri roches)
     - ✓ Insertion
   - Tri rapide : QuickSort
9. Introduction aux interfaces graphiques (Swing)
   - Notion de *container* et de *composants*
   - JFrame, JLabel, JTextField, JButton
   - Gestion d'événements (Listener)
     - ✓ ActionEvent (souris et clavier)
     - ✓ ItemEvent
     - ✓ Gestionnaire d'événements et leur enregistrement
   - Les GUI et les threads
   - Les boite de dialogue (classe JoptionPane)
   - JPanel, JTextArea, JRadioButton, JCheckBox
   - Notion de layout manager (BorderLayout, FlowLayout, GridLayout)
   - Exemple d'un projet d'interface intégrant les notions de base

RÉFÉRENCES

N R Séguin, Normand – *Recueil d'exemples pour INF2120*
Disponible à la Coop des Sciences

U R **http://www.grosmax.uqam.ca/prog/**
Site web du cours

V R LEWIS & CHASE – *Java Software Structures* – **2e édition, Addison-Wesley, 2005.**

V R DELANNOY, C. – *Programmer en Java* – **3e édition, Eyrolles, 2004.**

L R *Environnement de développement BlueJ (gratuit)* – **http://www.bluej.org**

A : article – C : comptes rendus – L : logiciel – N : notes – R : revue – S : standard – U : uri – V : volume

C : complémentaire – O : obligatoire – R : recommandé