|  |  |  |
| --- | --- | --- |
| **Plan de cours**  **Plan de cours** | | |
| **Numéro du cours** | **Pondération** | **Unités** |
| 420-EAB-LI | 3-4-3 | 3,00 |
| **H 15** | **Titre du cours** | **Département** |
| Programmation orientée objet II | Informatique |
| Logo-sans%20fond  **Campus Québec** | **Programme** | **Enseignant** |
| **AEC Programmeur- Analyste** | **Naji Bricha** |

**Situation du cours dans le programme**

Le cours **Programmation orientée objet II** (420-EAB-LI ) situé en deuxième session, est le second de quatre cours de programmation orientée objet offerts au niveau collégial en Informatique. Ce groupe de cours visent :

1. l’intégration de solides assises conceptuelles et méthodologiques en algorithmique et en programmation;
2. le développement d’habiletés opérationnelles en programmation orientée objet;
3. le développement d’habiletés fondamentales et d’attitudes professionnelles essentielles au travail en informatique de gestion : l’analyse logique, le raisonnement, la résolution de problèmes, la créativité, la volonté de comprendre en profondeur, le recours autonome aux ressources de l’environnement, la persévérance, la rigueur;
4. la mise en relation opératoire autonome des acquis informatiques et des objets traités en administration et en gestion.

Ces quatre cours sont :

* 420-EAA-LI Programmation orientée objet I, à la première session;
* 420-EAB-LI Programmation orientée objet II, à la deuxième session;
* 420-EAC-LI Programmation orientée objet III, à la troisième session;
* 420-EAD-LI Programmation IV, à la quatrième session.

En Programmation orientée objet I(420-EAA-LI ), les étudiants sont devenus capables de développer des solutions algorithmiques à des problèmes simples en utilisant un langage de programmation. Dans ce deuxième cours, **Programmation orientée objet II** (420-EAB-LI ), ils doivent acquérir les capacités d’organisation logique, de manipulation et d’exploitation de données nécessaires à la conception et à l’élaboration d’applications informatiques servant des fins en gestion ou en administration. Dans le cours Programmation orientée objet III(420-EAC-LI ), en troisième session, les étudiants s’appuieront sur les acquis des deux premiers cours pour construire des applications plus complètes au niveau de l’interface et la gestion des données.

Le cours se donne parallèlement avec les cours :

* Mathématiques de l’informatique, qui lui donnera des notions sur les systèmes de numérotations binaire, octal et hexadécimal, utilisés pour l’adressage sur un réseau informatique;
* Base de données I, qui lui permet de modéliser les données d’une situation fournie, implanter et exploiter les données relatives à cette solution à l'aide d’un langage structuré de requêtes;
* Architecture de réseaux I, qui se voit comme une introduction aux notions de base en réseautique et à l’utilisation et la configuration de certains services. Parmi ces services les étudiants auront l’occasion de travailler avec un serveur Web qui sera aussi utilisé par le cours *Programmation orientée objets II* avec les services des couches supérieures du modèle OSI.

Le cours Programmation orientée objet I(420-EAA-LI ) est préalable au cours **Programmation orientée objet II**(420-EAB-LI )*.* En effet, pour être admis dans le deuxième cours d’informatique, l’étudiant doit obtenir une note d’au moins 50% en Programmation orientée objet I. De plus pour être admis dans le troisième cours d’informatique Programmation orientée objet III (420-EAC-LI ), l’étudiant doit obtenir une note d’au moins 50% en Programmation orientée objet II.

**Énoncé des compétences associées au cours**

016TAppliquer une approche de développement par objets (Compétence partielle).

0171 Corriger des programmes (Compétence partielle).

**Objectif terminal du cours**

Au terme du cours, en faisant appel à un langage de programmation orienté objet, l’étudiant sera capable de programmer, selon les normes et le contexte, une application de complexité moyenne, en utilisant des données organisées en mémoire sous forme de vecteurs et de matrices. Ce cours va donc permettre à l’étudiant d’intégrer les apprentissages effectués pendant la première session et d’en améliorer la maîtrise de même que de perfectionner la résolution de problèmes à caractères génériques en abordant des problèmes plus complexes qu’à la première session. De plus, ce cours permet d’introduire les structures de données, de comparer et choisir les meilleures mises en œuvre des structures en fonction du problème à traiter; et de mettre en pratique les notions orientées objet plus avancées.

L’atteinte de cet objectif implique que l’étudiant sait :

* configurer et utiliser adéquatement un environnement de développement ;
* utiliser adéquatement un système de contrôle de version ;
* résoudre des problèmes plus complexes que ceux du premier cours;
* modéliser une application générique de moyenne complexité en utilisant une approche par objets ;
* organiser logiquement des données et les exploiter en mémoire en élaborant des algorithmes efficaces;
* écrire des programmes orientés objet fonctionnels et conformes aux normes de programmation et de documentation dans deux langages différents;
* transférer les connaissances acquises préalablement lors la production d’interfaces personne-système en créant des interfaces normalisées et conviviales ;
* porter un regard critique sur la démarche suivie et sur l’utilisation des structures mises à sa disposition en fonction du résultat voulu;
* déterminer les causes des problèmes de fonctionnement d'un programme et les corriger ;
* utiliser adéquatement une technique de tests structurés ;
* prendre en charge le développement de sa compétence et être en mesure d’identifier les aspects de sa compétence à développer ;

**Modalités d’évaluation des apprentissages**

|  |  |  |  |
| --- | --- | --- | --- |
| **Capacités essentielles à évaluer de façon sommative** | **Aspects évalués et critères d’évaluation** | **Moyens (dispositifs) privilégiés** | **Pondération** |
| **Capacité 1 :**  Écrire des programmes orientés objet fonctionnels et conformes aux normes dans 2 langages différents. | Établissement des relations hiérarchiques appropriées entre les classes.  Détermination juste des services des classes liées par l’héritage.  Programmer adéquatement une classe en utilisant différentes techniques de contrôle de sécurité et des erreurs.  Programmer adéquatement des classes liées par l’héritage.  Programmer des applications simples utilisant les comportements intrinsèques des classes liées par l’héritage et le polymorphisme. | Examens (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative.  Laboratoires (individuels) – en environnement non contrôlé et sont évalués à au moins 2 reprises de façon sommative.  Quiz (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative. | 40% |
| **Capacité 2 :**  Créer des structures de données simple en mémoire (vecteurs et matrices) dans 2 langages différents. | Analyse juste du contexte d’utilisation des données.  Choix pertinent de la structure de données appropriée au problème (vecteurs et matrices).  Utilisation juste des éléments du langage pour construire des structures de données en mémoire.  Utilisation juste des structures de contrôle du langage pour initialiser et parcourir les structures de données.  Adaptation correcte des classes faisant l’objet d’élément de données dans une structure de données en termes d’affichage.  Gestion des erreurs lors de la création, initialisation et parcourt d’une structure de données simple. | Examens (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative.  Laboratoires (individuels) – en environnement non contrôlé et sont évalués à au moins 2 reprises de façon sommative.  Quiz (individuel) – en environnement contrôlé et est évalués à au moins une reprise de façon sommative. | 20% |
| **Capacité 3 :**  Organiser et manipuler des données emmagasinées dans des structures données simples en mémoire (vecteurs et matrices) dans 2 langages différents. | Adaptation correcte des classes faisant l’objet d’élément de données dans une structure de données en termes de comparabilité.  Mise à jour correcte des structures de données en fonction de certaines opérations classiques (tri, ajout, modification, suppression, fusion, etc.)  Gestion des erreurs lors de la manipulation des données à partir d’une structure de données.  Détermination et utilisation justes de certaines structures de données fournies par le langage (liste et vecteur dynamique). | Examens (individuels) – en environnement contrôlé et sont évalués à 2 reprises de façon sommative.  Laboratoires (individuels) – en environnement non contrôlé et sont évalués à au moins 2 reprises de façon sommative.  Quiz (individuels) – en environnement contrôlé et sont évalués à au moins 2 reprises de façon sommative. | 40% |

Pour réussir le cours, l’étudiant doit démontrer sa maîtrise des trois capacités énumérées ci-haut (une note de 60% ou plus pour chacune des capacités). Dans le cas où l’étudiant n’a pas démontré sa maîtrise d’au moins l’une des capacités associées au cours, une note maximale de 55% lui sera attribuée pour le cours et celui-ci sera considéré comme un échec.

**Structure du cours, déroulement de l’apprentissage et de l’enseignement, activités nécessaires**

Le cours comporte deux volets : l’un orienté principalement vers l’intégration des concepts théoriques, l’autre, en parallèle, vers la maîtrise de l’application de ces concepts.

Les premières semaines servent à assimiler et à mettre en application les concepts de l’orienté objet (classes, héritage, surcharge, polymorphisme, interface) tout en abordant ou en revisitant des structures de données de tableau dynamique, de matrices et de vecteur statiques. Le cœur de la session est consacré à apprivoiser et à assimiler les concepts, les méthodes et les techniques de structuration de données d’objets plus complexes, à savoir les tableaux à une et deux dimensions et les collections, dans des contextes de tri, de recherche et de fusion.

Les dernières semaines seront consacrées à l’application des concepts déjà vus dans un environnement de type Web en PHP. En théorie, les notions peuvent être rapidement intégrées par des exercices formatifs sur ordinateur (dual work). En laboratoire, l’étudiant utilise des diagrammes et des algorithmes qu’il conçoit en utilisant les notions théoriques apprises en classe. Cela lui permet de résoudre des problèmes moyennement complexes et d’en faire la programmation et la mise au point.

Les stratégies suivantes de formation seront utilisées : animation d’ateliers théoriques, démonstration d’applications développées, exercices divers mettant en pratique les notions récemment apprises, consultation de l’aide en ligne, réalisation d’applications de plus en plus complexes qui respectent des normes de qualité. À cet égard, certaines parties du code seront déjà fournies à l’étudiant afin de lui permettre de voir graphiquement le résultat de son application.

**Attention :**

Effectuez à chaque semaine les 3 heures prévues comme **travail personnel** afin de réviser la théorie, lire les documents, étudier, **faire vos travaux**. Révisez fréquemment vos notes de cours.

Planifiez votre travail de façon à ne pas être en retard; appliquez toujours les normes de programmation vues en classe. Enfin, comprenez bien vos programmes puisque les examens vérifient, entre autres choses, votre compréhension de ces programmes.

Les micro-ordinateurs utilisés sont des compatibles IBM PC avec le système d’exploitation Windows 8. Le langage de programmation est le JAVA et PHP. L’environnement de développement est Eclipse.

**Échéancier prévu**

|  |  |  |
| --- | --- | --- |
| **Sem.** | **Théorie** | **Laboratoire** |
| 1 | Plan de cours  Révision des normes de programmation  Révision classes d’objets :  Diagrammes de classes et constructeurs, get, set, validation  Révision interface graphique en javaFX | Exercices  (Révision générale) |
| 2 | JavaFX Dialogs  Tableaux 2D de type primaire et d’objets | Laboratoire 1 (3 semaines) |
| 3 | Tri, fusion et recherche de données de type primaire |
| 4 | Les fichiers |
| 5 | Les fichiers  Exceptions  Héritage et polymorphisme | Laboratoire 2 (3 semaines) |
| 6 | Héritage et polymorphisme  Classes abstraites  Redéfinition de méthodes |  |
| 7 | **Examen 1 (25 %)** |  |
| 8 | Héritage et polymorphisme  Interfaces | Laboratoire 3 (3 semaines) |
| 9 | Environnement de programmation  PHP en ligne de commandes  Introduction au PHP et parallèle avec Java  Révision des notions de serveur Web |
| 10 | Tableaux 1D, 2D et associatifs  PHP et le Web |
| 11 | POO en PHP (Exception, Héritage, polymorphisme et surcharge de méthodes) | Exercices  PHP et interface web |
| 12 | Formulaires web | Laboratoire 4 (PHP et le Web) (3 semaines) |
| 13 | Formulaires web  PHP et le web |
| 14 | PHP et le web |
| 15 | **Examen 2 (30 %)** |  |

**Tableau des capacités versus les activités d’évaluation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Tp#1** | **Tp#2** | **Exa#1** | **Tp#3** | **Exa#2** | **Tp#4** | **Quiz (1 à 5)** |
| **Capacité #1** |  |  |  |  |  |  |  |
| **Capacité #2** |  |  |  |  |  |  |  |
| **Capacité #3** |  |  |  |  |  |  |  |

**Évaluation**

|  |  |
| --- | --- |
| Examen 1 | 25 % |
| Examen 2 | 30 % |
| 5 Quiz | 10 % |
| **Total examens** | **65 %** |
| Travail pratique 1 | 8 % |
| Travail pratique 2 | 8 % |
| Travail pratique 3 | 10 % |
| Travail pratique 4 | 9 % |
| **Total travaux** | **35 %** |

Jusqu'à 10% de la note totale d'un examen ou d'un laboratoire pourra être enlevé pour les erreurs de français (orthographe ou syntaxe).

Pénalité de 10% par jour pour la remise d'un travail en retard. Lorsque vous remettez un travail dans mon casier, vous devez le faire signer par un professeur du département qui indiquera la date et l'heure de la remise. Sinon la date de remise sera celle à laquelle je l'aurai recueilli. Vous avez la responsabilité de produire l’impression de vos travaux.

Tous les travaux pratiques (laboratoires) doivent être faits individuellement à moins d’indication contraire. Conformément à la PIÉA, « le plagiat, la tentative ou la collaboration au plagiat entraîne la note 0 pour l’activité d’évaluation concernée et élimine tout droit de reprise »

**Matériel obligatoire pour le cours**

**Notes de cours et les exemples de code seront disponibles sur le réseau.**

**IMPORTANT** : Un support amovible (clé USB, disque USB, …) vous permettant de conserver toujours une **deuxième copie (backup)** de vos travaux.

**Médiagraphie**

API Java

<http://docs.oracle.com/javase/7/docs/api/>

JavaFX

<http://docs.oracle.com/javase/8/javafx/api/toc.htm>

<http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm>

<http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html>

<http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>

Programmation Java

<http://java.developpez.com/cours/>

<http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java>

DOUDOUX, JEAN-MICHEL, Développons en Java, PDF, 2011, 1990 pages, <http://jmdoudoux.developpez.com/cours/developpons/java/>

<http://java.developpez.com/livres/#Debutants>

IDE

<http://eclipse.developpez.com/cours/>

API PHP

<http://www.php.net>

Programmation PHP

<http://php.developpez.com/cours/>

<http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql>

|  |
| --- |
| Modalités départementales d'évaluation des apprentissages |
| En conformité avec la politique institutionnelle d’évaluation des apprentissages (P.I.É.A) version août 2010, le département d’informatique apporte les précisions suivantes:   |  |  | | --- | --- | | Article 56  L’évaluation de l’expression et de la communication en français | Les enseignants du département d’informatique évaluent la qualité de la langue française. Cette évaluation représente 10% de toute évaluation sommative. | | Article 64  Remise en retard d’une réalisation servant à l’évaluation sommative | À l’exception des cas de force majeure, une remise en retard des travaux entraîne automatiquement une pénalité de 10% par jour de retard.  Une fois qu’une réalisation servant à l’évaluation est corrigée et remise aux étudiants, l’étudiant qui n’a pas encore remis son travail se voit attribuer la note zéro à cette activité d’évaluation. | | Article 67  La procédure de révision de note en cours de session | L’étudiant qui veut faire une demande de révision de notes partielle doit d’abord s’adresser à son enseignant ou enseignante. S’il est insatisfait du résultat de sa démarche, il s’adresse alors au coordonnateur du département. Celui-ci dirigera un comité formé de l’enseignant concerné et d’un enseignant qui possède les compétences à juger les motifs du litige. Le résultat devra être communiqué à l’étudiant dans les 10 jours ouvrables suivant sa demande. | | Article 27  Le plan de cours  b) L’adoption des plans de cours | Les plans de cours sont analysés par des comités sessions regroupés, selon le cas, par voie de sortie et cours de service. Pour chaque cours, une grille d’analyse, adoptée en département, est remise à la coordination. Une fois révisés par un comité, les plans de cours sont adoptés un à un en département où l’on souligne les nouveautés et faits saillants de chaque plan de cours. Cette procédure est complétée avant le début de la session concernée. | | Article 68  La procédure de révision de la note finale | Le comité de révision de notes est constitué de trois (3) enseignants y compris le responsable du cours dont l’évaluation est contestée. Si possible, le comité est formé d’enseignants qui font partie du comité du cours concerné. L’étudiant peut venir exposer sa demande et son point de vue conformément aux conditions stipulées dans la PIEA. |   *Adopté par le département d’informatique, le 27 août 2013* |