420.B0 TECHNIQUES DE L'INFORMATIQUE

Discipline principale : Département des Techniques de l'informatique

Spécialisation :
✓ Développement d'applications informatiques

420 3C6 MO – STRUCTURES DE DONNÉES

Pondération	Unités	Heures d'enseignement	Session
3-3-3	3	90	3

Description du cours
Ce cours permettra à l'étudiante ou à l'étudiant d'écrire, d'appliquer et de mettre en œuvre des programmes plus complexes en utilisant différentes structures de données. Il vérifiera différentes façons de structurer les données.

Objectif intégrateur
Exploiter les principes avancés de la programmation orientée objet.

Objectif ministériel
00Q6 (éléments 1 à 7) : Exploiter les principes de la programmation orientée objet.

Objectifs d'apprentissage
1. Appliquer des techniques de programmation avancées pour résoudre des problèmes.
2. Utiliser des structures des données avancées.
3. Mettre en œuvre la programmation en fonction des structures de données.
4. Vérifier la qualité de la programmation.

Attitudes professionnelles
❒ Rigueur
❒ Autonomie

Habiletés transdisciplinaires
✓ Compétences langagières : déterminer l'information à rédiger et effectuer une notation claire du travail, tout en utilisant la terminologie appropriée à la profession.
✓ Profil Tic : exploiter les TIC de manière efficace et responsable; rechercher, traiter et présenter de l'information.
✓ Risques en matière de santé et de sécurité au travail : risques chimiques ou dangers d'ordre chimique; risques physiques ou dangers d'ordre physique; risques ergonomiques ou dangers d'ordre ergonomique; risques psychosociaux ou dangers d'ordre psychosocial; maintenir un environnement physique de qualité.
✓ Entrepreneuriat : travailler à son compte en développement d'applications ou de réseaux; créer un bureau de consultation.

Spécialisation Développement d'applications informatiques
Ces cours sont préalables absolus au présent cours
420 ZD5 MO Programmation orientée objet
420 ZF5 MO Programmation structurée
Le présent cours est préalable absolu aux cours ci-après
420 4E6 MO Analyse et conception de modèles
420 5G4 MO Technologies émergentes des applications
420 6AM MO Stage en développement d'applications

CONTEXTE PARTICULIER D'APPRENTISSAGE

En laboratoire informatique.

FICHE TECHNIQUE

Ordinateur équipé de deux écrans (un étudiant par ordinateur), projecteur multimédia, accès à Internet, accès à une imprimante.

Logiciels requis, 1 copie par étudiant :

- Version récente du langage de programmation utilisé (p.ex. le l'ensemble de développement Java).
- Version récente d'un logiciel d'environnement de développement intégré (p.ex. Eclipse).
- Version récente d'un logiciel de gestion de versions (p.ex. Git).

Technicienne ou technicien en travaux pratiques.

ACTIVITÉ D'ÉVALUATION FINALE DU COURS

L'étudiant écrira des programmes plus complexes en appliquant des techniques de programmation avancées et en utilisant différentes structures de données.

FICHE TECHNIQUE	
Forme	Travaux pratiques.
Contexte de réalisation	• À partir d'un problème et à l'aide de règles de nomenclature et de codage.
Pondération	Entre 30 % et 40 %

CRITÈRES D'ÉVALUATION
Représentation correcte des algorithmes.
Détermination juste des classes à modéliser.
Choix approprié des instructions, des types de données élémentaires et des structures de données.
Intégration correcte des classes dans le programme.
Fonctionnement correct du programme.

Fiche d'intégration des objectifs du cours 420 3C6 MO – STRUCTURES DE DONNÉES

Objectif intégrateur

Exploiter les principes avancés de la programmation orientée objet.

Objectifs d'apprentissage

Appliquer des techniques de programmation avancées pour résoudre des problèmes.
(00Q6 : élément 3)
IMPORTANCE APPROXIMATIVE RELATIVE : 25 %

Utiliser des structures des données avancées.
(00Q6 : éléments 1 et 2)

IMPORTANCE APPROXIMATIVE RELATIVE : 40 %

Mettre en œuvre la programmation en fonction des structures de données.
(00Q6 : éléments 4, 5 et 6)
IMPORTANCE APPROXIMATIVE RELATIVE : 20 %

Vérifier la qualité de la programmation.
(00Q6 : éléments 5 et 7)

IMPORTANCE APPROXIMATIVE RELATIVE : 15 %

Contenus essentiels

- Récursivité.
- Algorithmes de tri : tri échange, tri insertion, tri sélection, tri à bulles (*bubblesort*), tri shell (*shellsort*), tri rapide (*quicksort*), tri fusion (*mergesort*).
- Utilisation des classes du langage de programmation pour manipuler des tableaux (p.ex. *Arrays*).
- Gestion de tableaux dynamiques (à taille variable).
- Utilisation des classes du langage de programmation pour manipuler des tableaux dynamiques (p.ex. *ArrayList*, *Vector*, *ArrayDeque)*.
- Algorithmes de mise à jour d'un fichier d'objets. Algorithme d'ajout, de modification et de suppression des objets dans un fichier.
  - Technique de l'accès séquentiel.
  - Technique de l'accès direct,
  - Technique de l'accès indexé.

- Définition et implémentation de listes chaînées.
- Utilisation des classes du langage de programmation pour gérer des listes chaînées (p.ex. *LinkedList*, *Collections*).
- Définition et implémentation de piles.
- Utilisation des classes du langage de programmation pour gérer des piles (p.ex. *Stack*, *ArrayDeque*, *LinkedList)*.
- Définition et implémentation de files.
- Utilisation des classes du langage de programmation pour gérer des files (p.ex. *ArrayDeque*, *LinkedList)*.
- Définition d'autres sortes de listes : listes circulaires, symétriques, symétriques circulaires, multilistes, listes indexées.
- Définition et implémentation d'arbres binaires.
- Utilisation des classes du langage de programmation pour gérer des arbres binaires (p.ex. *TreeSet*, *TreeMap*).
- Adressage dispersé (hachage ou *hashing*) : fonctions de hachage, traitement des collisions.
- Utilisation des classes du langage de programmation pour gérer l'adressage dispersé (p.ex. *HashSet*, *LinkedHashSet*, *HashMap*, *LinkedHashMap)*.
- Modèles de conception (design patterns) : définition, modèles de création, de structure et de comportement.

- Programmation des classes du programme.
- Choix des instructions, des types de données élémentaires et des structures de données.
- Intégration des classes dans le programme.
- Génération de l'interface graphique du programme.
- Documentation du programme.

- Vérification du fonctionnement du programme : jeux d'essai, repérage complet des erreurs de fonctionnement.
- Respect des normes de programmation.
- Utilisation d'un outil de gestion des versions de programmes (p.ex. *Git*).

Activité de rétroaction

Activité de rétroaction

Activité de rétroaction

Activité de rétroaction

Activité d'intégration

Activité d'évaluation finale du cours