LE GOUVERNEMENT
DU GRAND-DUCHÉ DE LUXEMBOURG
Ministère de l'Éducation nationale,
de l'Enfance et de la Jeunesse

Enseignement secondaire général
Classes supérieures
Division technique générale
Section informatique
PROGR / Programmation
Programme
1GIN

Langue véhiculaire :	français
Nombre de leçons :	4
Nombre minimal de devoirs par semestre :	2
Dernière mise à jour par la CNES :	24/05/2021

Remarques générales

1. Vu l'aspect pratique du cours, celui-ci doit avoir lieu entièrement dans une salle d'informatique avec un poste par élève. L'utilisation d'une tablette ne se substitue pas à un ordinateur dans ce cadre.
   Le nombre d'élèves par auditoire ou par enseignant doit être limité à 15.
   Il est recommandé d'organiser le cours à raison de blocs de deux leçons consécutives.
2. Ce cours peut être suivi et conclu avec succès uniquement si les élèves peuvent se préparer et s'exercer sur un ordinateur de bureau en dehors des cours. L'accès à un ordinateur qui permet d'installer le JDK Java (donc tournant sous MacOS, Windows ou Linux) est nécessaire.
3. Le programme est axé sur quelques projets définis, dans le cadre desquels les élèves apprennent à manipuler les différents éléments énumérés dans la partie "Détails".
4. Les applications à interface graphique développées dans le cours ne se limitent pas nécessairement à un seul formulaire.
5. Les algorithmes plus complexes sont développés et représentés sous forme de structogrammes. Les élèves doivent être capables de lire et interpréter les structogrammes ainsi que de réaliser les algorithmes correspondants en Java.

A. Objectif

L'élève est capable de concevoir, de réaliser et d'adapter des applications sous un environnement graphique et événementiel selon le modèle ***MVC*.** A cet effet il connaît les manipulations essentielles du langage de programmation et fait appel à la pensée informatique pour résoudre des problèmes sous forme de projets.
En vue d'études supérieures en informatique, l'élève connaît les algorithmes et structures de données usuelles en informatique et les techniques de la programmation orientée objet moderne. Il sait organiser les données et le code de façon efficace. Le cours n'a ni l'envergure ni le rythme d'un cours post-secondaire, mais il prépare l'élève aux réflexions qui l'aideront à pouvoir suivre des études supérieures avec succès.
Pour un problème donné d'envergure moyenne (c.-à-d. qui se laisse résoudre en moins de 12 heures de travail), l'élève sait analyser le problème et établir une structure de classes appropriée en évitant la duplication de code et en appliquant correctement les techniques de la POO (encapsulation, héritage, polymorphisme, ...).

Compétences ciblées	Contenus	Durée
1. Savoir employer les fonctions avancées sur les chaînes de caractères	algorithme d'extraction d'une sous-chaîne délimitée par un symbole donné	1
2. Connaître la notion d'exception	- Le principe 'catch-or-throw' - structure *try … catch* - mot clé *throws*	3
3. Être capable de lire et écrire dans des fichiers textes et binaires	- classes *FileOutputStream*, *FileInputStream*, *BufferedInputStream*, *BufferedOutputStream*, *DataInputStream*, *DataOutputStream*, ObjectInputStream, *ObjectOutputStream*, *FileReader*, *FileWriter*, *BufferedReader*, PrintWriter, *Scanner* - l'interface *Serializable* - *JFileChooser* et son emploi - Lire et écrire des fichiers XML et JSON à travers une bibliothèque prédéfinie (p.ex *XStream* ou *gson*) - nommer et localiser les fichiers sur le disque	6
4. Savoir employer les interfaces existants les plus importants et savoir définir de nouveaux interfaces simples	- les interfaces existants les plus importants (*Iterable*, *Cloneable*, *Comparable*, ...) - utilisation des interfaces existants - analyse de la réalisation de méthodes de réaction en NetBeans *(Listener, Adapter*) - implémentation d'un itérateur - définition et utilisation d'un nouvel interface dans le cadre d'un problème donné - le schéma *Observer-Observable* et sa réalisation par *PropertyChangeListener* et *PropertyChangeSupport*	16
5. En connaissance des algorithmes récursifs de base, savoir employer la récursivité pour réaliser des algorithmes efficaces	étude d'algorithmes récursifs standard et de leur efficacité, notamment : - factorielle, - nombres de Fibonacci, - tours de Hanoi, - recherche dichotomique, - le tri *'quicksort',* - dessin de fractales, - évaluation d'expressions simples, - détermination du chemin le plus court	22
6. Connaître les caractéristiques et savoir employer les collections prédéfinies les plus importantes	- les classes génériques et leur emploi - étude des collections standard et de leur caractéristiques : *ArrayList*, *LinkedList*, *TreeSet*, *TreeMap*, *HashMap*	8

Compétences ciblées	Contenus	Durée
	- comparaison de l'efficacité des différentes structures en pratique - le problème de la synchronisation et sa résolution	
7. Savoir organiser et traiter les données à l'aide des structures de données standard	étude des structures standard suivantes et application à des exemples concrets : - tableaux *(array)* à une dimension - tableaux (*array*) à deux dimensions - les types d'énumération *(enum)* - piles (*stacks*) et files d'attente (*queues*) - listes chaînées d'objets - arbres binaires de recherche - adressage dispersé (*hashing*) - tables d'association clé-valeur (*HashMap, TreeMap*) - définition de collections simplifiées (*Linked List*, *BST*, *HashMap*) génériques et non génériques	36
8. Savoir organiser et structurer les classes, attributs et méthodes en appliquant les principes de la POO de façon appropriée	- structure générale - encapsulation et symboles de visibilité - relations (héritage et agrégation) - héritage, polymorphisme, *late-binding*	8
	TOTAL :	**100**

B. Détails

Package	Classe	Details
java.awt.event	ActionListener	
java.util	Scanner	Constructor : Scanner(...) Méthodes - next() et hasNext() - nextBoolean(), nextDouble(), nextInt() - nextLine()
	ArrayList, LinkedList, TreeSet, HashMap, TreeMap,	Méthodes ajout, recherche, suppression d'éléments, obtention et itération de tous les éléments, de leurs valeurs et de leurs clés
java.lang	Throwable et descendants (Exception, Error, ...)	Méthodes - getMessage() - printStackTrace() - toString()
	String	Méthodes - equals(), compareTo(...) - contains(...) - indexOf(...) - substring(...) - valueOf() - toLowerCase(), toUpperCase() - replace(...) - trim() - charAt(...) - split(...)
javax.swing	JFileChooser	Constructor : JFileChooser(...) Méthodes - showSaveDialog(...), showOpenDialog(...) - getSelectedFile(), getName(...) - addChoosableFileFilter(...), setAcceptAllFileFilterUsed(...)
javax.swing. filechooser	FileFilter et FileNameExtensionFilter	Constructor : FileNameExtensionFilter(...)
java.io	FileInputStream et FileOutputStream	Constructors : FileInputStream(...) et FileOutputStream(...)
	BufferedInputStream et BufferedOutputStream	Constructors : BufferedInputStream(...) et BufferedOutputStream(...)
	DataInputStream et DataOutputStream	Constructors : DataInputStream(...) et DataOutputStream(...) Méthodes : - read...() et write...() pour différents types - close()
	ObjectInputStream et ObjectOutputStream	Constructors : ObjectInputStream(...) et ObjectOutputStream(...) Méthodes : - readObject(...) et writeObject(...) - close()
	FileReader et FileWriter	Constructors : FileReader(...) et FileWriter(...) Méthodes - write(...) de FileWriter - close()
	BufferedReader	Constructor : BufferedReader(...) Méthodes : read(...) et readLine()
	PrintWriter	Constructor : PrintWriter(...) Méthodes : print(...) et println()

Package	Classe	Details
java.beans	PropertyChangeSupport et PropertyChangeListener	Constructors : PropertyChangeSupport(...), Méthodes : - propertyChange(...) - addPropertyChangeListener(...), removePropertyChangeListener(...) - firePropertyChange(...)

C. Logiciels à utiliser :

Environnement de développement :

- Unimozer (http://unimozer.fisch.lu)
- NetBeans (http://netbeans.org)

D. Liens importants

- Ressources concernant le cours d'introduction à la programmation orientée objet
    - http://java.cnpi.lu/
- JavaDoc
    - https://docs.oracle.com/en/java/javase/
    - http://docs.oracle.com/javase/tutorial

E. Évaluation trimestrielle :

Les devoirs en classe peuvent se présenter sous différentes formes :

- devoir pratique sur ordinateur,
- devoir écrit,
- projet ou partie d'un projet,
- contrôle continu (maximum 1/6 de la note semestrielle).