Sujet du projet

Site: LMS UCA
Cours: SLUIN502 - UE Programmation et conception orientee objet
Livre: Sujet du projet

Imprimé par: Quentin beauchet
Date: samedi 9 janvier 2021, 00:16

Table des matières

1. Les données à manipuler

2. Représentation

3. Travail demandé

3.1. Jalon 1

3.2. Jalon 2

3.3. Jalon 3

3.4. Jalon 4

4. Aide technique

4.1. Lecture d'un fichier XML

5. FAQ

5.1. Les fichiers XML

5.2. La modélisation

5.3. Fonctionnalités demandées

1. Les données à manipuler

Un **cours** est l'unité d'enseignement de base d'une université. Il est identifié par un *code unique*. Chaque cours dispose d'un *nom* et est associé à un certain nombre de *crédits*.

Un **bloc** représente une unité d'enseignement plus souple que le cours. Comme les cours, ils sont identifiés par un *code unique* et dispose d'un *nom* et d'un nombre de *crédits*. Il en existe plusieurs types :

- *Un **bloc simple*** est juste un cours.
- *Un **bloc à options** est composé de plusieurs cours portant le même nombre de crédits. C'est aussi le nombre de crédits du bloc.*
- *Un **bloc composite** est composé de plusieurs cours. Le nombre de crédits du bloc est la somme des crédits des UE qui le composent.*

Un **programme** est une *liste de blocs* à obtenir pour valider une étape. Chaque programme est identifié par un *code unique* et un *nom*.

Un **étudiant** est caractérisé par un *identifiant,* son *nom* et son *prénom*. Un étudiant peut être inscrit à (au plus) un *programme*. Il dispose également de *notes* dans des UE, indépendamment du programme qu'il suit.

Une **note** est soit la valeur ABI pour signaler l'absence aux examens soit une valeur numérique comprise entre 0 et 20.

Pour **valider un programme**, il faut obtenir la moyenne pondérée sur les blocs qui le composent. Le poids de chaque bloc est le nombre de crédits qu'il porte.

- La note d'un bloc simple est la note du cours associé.
- La note d'un bloc à options est le maximums des notes obtenues sur les cours qui le composent.
- La note d'un bloc composite est la moyenne pondérée des blocs qui le composent.

2. Représentation

L'ensemble des données est obtenu à l'aide d'un fichier XML. Le fichier se structure selon la grammaire suivante :

```
start          ::= preamble data
preamble       ::= <?xml version="1.0"?>
data           ::= <data> data_item* </data>
data_item      ::= course | program | student

course         ::= <course> identifier name credits </course>

program        ::= <program> identifier name bloc* </program>
bloc           ::= simple_bloc | option_bloc | composite_bloc
simple_bloc    ::= item
option_bloc    ::= <option> identifier name item+ </option>
composite_bloc ::= <composite> identifier name item+ </composite>

student        ::= <student> identifier name surname program? grade* </student>
grade          ::= <grade> item value </grade>

identifier     ::= <identifier> string </identifier>
name           ::= <name> string </name>
surname        ::= <surname> string </surname>
credits        ::= </credits> integer <credits>
item           ::= <item> string </item>
program        ::= <program> string </program>
value          ::= <value> ABI | double </value>
```

Un exemple de fichier vous est proposé pour effectuer vos tests.

3. Travail demandé

Ce projet est à réaliser à plusieurs. Vous devez former un groupe de 3 étudiants pour le mener à bien. Des groupes de 2 personnes sont autorisés si une bonne justification est apportée.

Vous devrez déposer votre travail dans la boîte de dépôt du projet. Ne faites qu'un rendu par groupe : une seule personne s'en charge. La date limite est fixée au 8 janvier 23h59.

Vous devez remettre l'ensemble du code source réalisé (pas les classes compilées ni les fichiers liés à votre IDE) et un rapport. Ce dernier devra couvrir a minima les points suivants :

- Les noms des membres du groupes (ça peut paraître bête mais j'ai déjà vu des oublis).
- Les fonctionnalités implémentées.
- La contribution de chacun dans le projet. La façon dont vous avez organisé votre travail et comment vous vous l'êtes partagé doit apparaître clairement.
- Les choix de conception réalisés. Ici, soyez concis et précis. Il est inutile de recopier et paraphraser tout votre code.
- Les difficultés rencontrées et les limitations ou bugs connus de votre programme.
- Les détails techniques pour lancer votre application. Ce n'est pas à moi de chercher votre `main` ni de choisir lequel lancer si vous en avez plusieurs. Si vous exploitez la ligne de commande, il faut l'indiquer.
- Tout autre information que vous jugerez pertinente.

3.1. Jalon 1

Dans ce premier jalon, on vous demande de produire un procès-verbal (PV) d'un programme en vue d'un jury.

Un PV sera donné sous la forme d'un fichier CSV. La première ligne est une ligne d'entête qui donne le nom de chaque colonne :

- N° Étudiant.
- Nom.
- Prénom.
- Le nom du programme.
- Le nom de chaque bloc du programme immédiatement suivi du nom des cours constitutifs s'il y a lieu.

Les lignes suivantes listent les étudiants inscrit dans le programme choisi avec leurs notes correspondantes. Les quatre dernières lignes donnent des indications statistiques sur les notes de la promotions : note min, max et moyenne ainsi que l'écart-type pour chaque colonne numérique.

Des exemples de fichiers CSV obtenus avec le jeu de données fourni sont disponibles pour les programmes L3I et L3MI.

Comme certains tableurs sont tatillons sur l'import des fichiers CSV (problèmes de localisation), voici une capture d'écran du rendu d'un tel fichier.

	A	B	C	D	E	F	G	H	I	J
1	N° Étudiant	Nom	Prénom	SLINF3 180 - L3 Informatique	SLUIN501 - Automates et langages	SLUIN502 - Programmation et conception orientées objet	SLUIN503 - Programmation fonctionnelle	SLUIN601 - Algorithmique 2	SLUIN602 - Compilation	SLOIN501 - OPTION 1 S5 INF
2	21781843	Agosti	Alexandre	12,262	16,903	14,159	11,202	8,985	13,591	10,246
3	21527159	Alphazan	Francois	13,58	11,238	12,694	15,156	16,088	12,787	15,747
4	21206751	Armanet	Richard	11,431	12,456	13,341	10,393	12,45	11,869	10,03
5	21151305	Attar	Emile	13,063	14,761	13,742	11,991	13,006	11,044	15,002
6	21172882	Azema	Frederic	12,666	12,391	10,153	15,35	11,106	11,497	17,034
7	21504386	Baltimore	Cecile	9,828	12,192	11,405	8,433	11,486	8,784	7,563
8	21037687	Baudin	Geraldine	7,822	12,241	6,741	9,458	12,348	8,485	8,029
9	21396583	Bazin	Colas	10,604	9,216	12,893	10,13	10,043	10,669	11,564
10	21445854	Belhronari	Gilbert	13,115	13,744	11,979	15,471	13,167	13,024	9,52
11	21116358	Berissi	Cimar	6,131	4,701	9,842	4,748	10,16	2,905	9,575
12	21774004	Bienfait	Dominique	7,355	8,049	7,759	5,007	9,328	5,016	7,635
13	21781500	Blot	Joseph	10,437	14,152	8,066	14,214	10,888	13,93	8,87
14	21402056	Bouilland	Sarah	10,284	9,264	9,954	6,502	13,08	8,702	14,289
15	21233286	Bounab	Patrice	13,317	10,312	7,894	13,142	14,928	9,602	19,675
16	21624947	Bouquet	Marie	7,489	11,433	8,586	3,139	5,481	11,999	4,883
17	21413340	Bourdon	Denis	6,53	6,99	7,11	10,873	5,035	7,703	5,823
18	21784991	Brachet	Mehdi	13,828	12,163	12,518	17,478	10,905	16,761	14,7
19	21232189	Broucke	Sebastien	6,513	9,498	9,414	5,659	ABI	6,943	10,079
20	TRONQUÉ POUR VOIR LE DÉBUT ET LA FIN DU FICHIER									
21	21840895	Rousseau	Dominique	14,169	11,973	13,742	10,659	14,561	12,853	17,373
22	21536961	Salmon	Mohamed	7,449	9,69	7,522	12,557	9,581	7,908	3,578
23	21034551	Scarbonchi	Florence	13,047	11,47	6,703	15,733	15,901	10,095	12,944
24	21148815	Sillion	Lolita	13,485	12,316	13,372	15,735	11,797	13,67	14,116
25	21215296	Somme	Alain	12,77	15,229	17,582	13,586	11,179	12,65	7,551
26	21055871	Speccogna	Alexandra	10,227	9,084	11,116	11,281	6,137	8,175	14,432
27	21801689	Syda	Nicolas	11,847	12,535	14,915	13,194	9,858	7,093	11,841
28	21210964	Tanne	Simonne	7,287	5,824	11,236	8,284	8,021	5,981	9,494
29	21706218	Toulet Suberbie Cousy	Roberte	8,404	9,986	7,964	9,136	8,081	8,223	9,992
30	21428341	Trotin	Eutmane	9,338	12,928	9,439	10,492	7,898	13,191	ABI
31	21391300	Ullmann	Guy	13,261	12,618	12,918	11,407	14,726	16,368	12,228
32	21771840	Vaquez	Henri	6,993	4,035	7,323	5,552	7,499	9,201	8,424
33	21173664	Vernay	Clement	12,172	12,962	9,318	14,007	12,137	9,482	10,992
34	Note max			14,384	17,933	19,024	17,554	20	17,597	19,675
35	Note min			4,792	2,861	2,447	2,218	4,343	2,905	2,251
36	Note moyenne			10,316	10,64	10,366	10,442	10,534	10,347	10,477
37	Écart-type			2,504	3,468	3,34	3,366	3,17	3,279	3,435
38										

Les enjeux de cette partie sont :

- La représentation des données.
- Leur importation depuis le fichier XML.
- Leur traitement (calculs de moyennes et de statistiques).
- L'exportation des données traitées vers un fichier CSV.

3.2. Jalon 2

Le premier jalon est relativement strict dans ce que vous devez faire. Les formats d'entrée et de sortie sont spécifiés et vous n'avez que peu de libertés. À partir du deuxième jalon, vous allez devoir faire vos propres choix. Il s'agit de développer une interface graphique. Vous devrez vous soucier de l'ergonomie et de la convivialité de votre application.

Dans ce jalon, on demande les fonctionnalités suivantes :

- Sélectionner un programme et afficher les étudiants qui le suivent avec leurs notes. Il s'agit plus ou moins de refaire le PV du premier jalon dans votre interface.
- Sélectionner spécifiquement des programmes ou des blocs et n'afficher que les notes associées.
- Afficher toutes les notes d'un étudiant. Notez que dans le jeu de données fourni, chaque étudiant n'a de notes que dans un programme. Ce n'est pas une obligation. Un étudiant pourrait avoir des notes dans plusieurs programmes ou dans des cours qui n'apparaissent dans aucun programme.
- Avoir une vue hiérarchisée d'un programme pour en visualiser la structure.

Comme vous avez dû le remarquer, on ne vous demande que de visualiser les données dans ce jalon. Vous n'avez pas à les modifier.

3.3. Jalon 3

On cherche désormais à pouvoir ajouter des données via l'interface graphique. On veut pouvoir :

- Ajouter/Modifier/Supprimer une note d'un cours (et donc recalculer les moyennes où elle intervient).
- Ajouter/Supprimer un étudiant.
- Ajouter un cours ou un programme (interdiction de supprimer les cours ou programmes déjà présents). Maj du 13/12/2020.
- Sauvegarder les données modifiées au format XML. Elles devront être rechargées automatiquement à la prochaine ouverture de l'application.

3.4. Jalon 4

Vous avez désormais une application fonctionnelle. Il vous reste du temps ? Vous voulez aller plus loin ? Ce jalon est libre ! Vous pouvez ajouter autant de fonctionnalités que vous le souhaitez pour améliorer le projet. Il s'agit de points bonus (et plafonnés). Faites-vous plaisir mais n'en faites pas trop !

4. Aide technique

Ce chapitre contient différentes aides techniques pour le projet. Il s'étoffera en fonction des problèmes que vous soulèverez.

4.1. Lecture d'un fichier XML

Voici le schéma pour lire un fichier XML.

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class XMLTest {
    public static void main(String[] args) {
        try {
            File file = new File("some path");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(file); // ouverture et lecture du fichier XML
            doc.getDocumentElement().normalize(); // normalise le contenu du fichier, opération très conseillée
            Element root = doc.getDocumentElement(); // la racine de l'arbre XML

            // c'est parti pour l'exploration de l'arbre

        } catch (Exception e) {
            // oups, pas normal
        }
    }

    // Extrait la liste des fils de l'élément item dont le tag est name
    private static List<Element> getChildren(Element item, String name) {
        NodeList nodeList = item.getChildNodes();
        List<Element> children = new ArrayList<>();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) nodeList.item(i); // cas particulier pour nous où tous les noeuds sont des éléments
                if (element.getTagName().equals(name)) {
                    children.add(element);
                }
            }
        }
        return children;
    }
}
```

Lorsqu'on est sur une feuille de l'arbre, on peut obtenir le contenu de l'élément à l'aide de la méthode `getTextContent()`.

5. FAQ

Les différentes questions reçues sont reproduites et organisées dans les sous-chapitres qui suivent.

Dernière mise à jour le 13/12/2020.

5.1. Les fichiers XML

Problèmes liés à l'analyse du fichier xml.

Que fait exactement la méthode getElementsByTagName ?

Un fichier xml code un arbre. La grammaire utilisée est :

```
node ::= <tag> (node | text)* </tag>
tag  ::= string
text ::= string
```

L'appel `node.getElementsByTagName(name)` renvoie la liste (sous forme de `NodeList`) des descendants (directs ou indirects) de `node` dont le tag est `name`.

Pourquoi avoir défini la méthode getChildren ?

La méthode `getChildren` a deux avantages par rapport à `getElementsByTagName`. Seuls les enfants directs sont renvoyés. De plus, ils sont renvoyés sous forme de `List<Element>` qui est plus pratique à manipuler.

Comment fonctionne une NodeList ? Et une List<E> ?

Sans rentrer trop dans le détail, les méthodes à utiliser sont `getLength()` et `item(int)` pour `NodeList`. Elles correspondent à `size()` et `get(int)` pour une `List<E>`. Elles renvoient respectivement le nombre d'éléments dans la liste et un élément d'indice donné.

Prenez garde que la méthode `item` renvoie un `Node` qu'il faudra caster en `Element` pour le projet. Préférez l'utilisation de `getChildren` qui automatise et simplifie le travail.

Les collections étudiées à la fin du cours expliqueront toutes les subtilités des `List<E>`.

Que fait la méthode getTextContent ?

Elle renvoie le texte du nœud sur lequel elle est appelée. Le texte d'un nœud correspond à tout ce qui n'est pas inclus entre un chevron ouvrant et le chevron fermant correspondant. Dans le cadre du projet, il ne faut donc l'utiliser que pour les feuilles du fichier xml.

Exemple : Si `node` est l'objet de type `Node` correspondant au nœud xml

```
<node>
  text
  <child>child text</child>
  other text
</node>
```

alors `node.getTextContent()` renvoie la chaîne `"\n  text\n  child text\n  other text\n"`. Attention, les espaces et sauts de lignes sont compris.

Pourrait-on avoir un autre exemple ?

L'exécution du code suivant

▶ Détails

affiche

▶ Détails

5.2. La modélisation

Est ce que les matières composants un programme sont des matières liées uniquement au programme, c'est à dire que les matières sont créées en même temps que le programme, ou est ce que ce sont des matières prédéfinies que l'ont peu utiliser lors de la création d'un programme . Est-ce qu'on doit pouvoir créer des matières ? Ou juste créer des programmes ?

Les cours sont indépendants des programmes. Ils peuvent apparaître dans 0, 1, 2 ou plus programmes. Dans le jalon 3, on souhaite pouvoir également ajouter des cours. Le sujet a été mis à jour pour être plus clair à ce niveau.

Le programme est créé à partir de cours préexistants (éventuellement ajoutés juste avant).

Quant est-il d'un bloc ?

Un bloc simple est un cours, la réponse précédente s'applique. Comme les blocs à options et les blocs composites sont entièrement définis par les cours qui les composent, on peut considérer qu'un bloc est lié à un programme ou indépendant. L'impact est nul.

Dans le fichier xml, les blocs sont définis directement dans les programmes. Le bloc de compétences transversales est défini de manière identique dans chacun des deux programmes proposés. Vous êtes libre de traiter ce cas comme vous le souhaitez.

Comment fonctionnent les crédits ?

Les crédits servent uniquement à pondérer les notes lors du calcul de la moyenne d'une UE composite ou d'un programme.

Un étudiant peut-il avoir des notes dans plusieurs options d'un bloc optionnel ?

Le sujet l'autorise et cela peut arriver concrètement (si on mime ce qui se passe à l'université) dans certains cas particuliers (changement de maquettes, redoublement, réorientation, ...). Dans ce cas, la note du bloc est le maximum obtenu dans l'une des options.

Si un étudiant à ABI dans toutes les composantes d'un bloc à option ou d'un bloc composite, sa note au bloc est-elle 0 ou ABI ?

ABI.

Comment compte-t-on les notes manquantes ou les notes ABI dans le calcul de la moyenne et de l'écart-type ?

On ne les compte pas. Seuls les notes numériques interviennent.

Quelle formule faut-il utiliser pour calculer l'écart-type ? J'ai vu qu'il existe une formule spécifique dans le cas de données obtenues via une moyenne pondérée.

Pour un échantillon x de taille n et de moyenne μ, l'écart-type est $\sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2 - \mu^2}$. On ignore volontairement le fait que l'échantillon soit ou non obtenu par une moyenne pondérée.

5.3. Fonctionnalités demandées

Pour le jalon 1, doit-on ne générer un fichier CSV uniquement pour un programme spécifique ?

Oui, un procès-verbal est lié à un programme spécifique. Vous pouvez le voir sur les deux exemples de fichiers CSV proposés. La base de données mixe L3I et L3MI mais il y a un procès-verbal distinct pour chacun d'entre eux.

Pour le jalon 2, doit-on seulement afficher les notes des étudiants dans le/les programmes sélectionnés ?

L'affichage est basé sur des filtres. Un programme est un filtre particulier. Vous pouvez avoir un filtre par défaut qui consiste à tout afficher (peu utile dans un cas réel) et il vous faudra donner la possibilité de créer des filtres personnalisés.

Que doit-on afficher si un étudiants n'a pas de note dans un cours, bloc ou programme ?

Si un étudiant n'a pas de note, la case associée reste vide.

Qu'appelez-vous une vue hiérarchisée d'un programme ?

Un programme est composé de blocs. Un bloc peut être lui même composé de cours (blocs à options ou composite). Une vue hiérarchisée permet de visualiser ces compositions.

Voici deux exemples d'affichage basique possibles. Le premier est statique, le second dynamique.

▶ Détails

▶ Détails