

Nom et prénom :
Login examen :

Institut Paul Lambin

Session de Juin 2019

Examen de Structures de données : Avancé

Christophe Damas, José Vander Meulen

Année(s) d'études : 2^{ème} année de baccalauréat en informatique de gestion

Date et heure : mercredi 29 mai à 9h00

Durée de l'examen : 3 h ; pas de sortie durant les 60 premières minutes

Contenu

1.	Questions sur machine	2
a)	Bulletins [8 pts]	3
b)	Graphe [5 pts]	6
c)	Récursion [1 pt]	7
d)	Huffman [2pts]	7
e)	Football [3 pts]	8
2.	Question sur papier [1 pt]	9

Total :	/20
----------------	------------

Nom et prénom :

Login examen :

1. Questions sur machine

Avant de commencer cette partie, suivez les étapes suivantes :

1. Dézippez le fichier compressé (**Extract here**)
2. Renommez tous les noms et prénoms des différents répertoires. Ex :
bulletins_DAMAS_CHRISTOPHE. (Evitez les caractères spéciaux : accents, ...)
3. **Utilisez la bonne version d'Eclipse (JEE) : L'eclipse normal ne contient pas le plugin XSL.**
4. Switchez votre workspace Eclipse afin qu'il soit positionné à la racine du Z.
5. Importez les différents projets dans votre workspace

Pour cette partie, on doit avoir dans le Z :

- **bulletins_NOM_PRENOM**
- **graphe_NOM_PRENOM**
- **recursion_NOM_PRENOM**
- **huffman_NOM_PRENOM**
- **football_NOM_PRENOM**

Nom et prénom :

Login examen :

a) Bulletins [8 pts]

Dans le répertoire bulletins, vous trouverez un schéma XML valide intitulé bulletins.xsd.

Dans un document valide selon ce schéma, une école enregistre les notes d'examen des étudiants de chaque bloc (bloc 1, bloc 2 et bloc 3).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="note">
    <xs:restriction base="xs:positiveInteger">
      <xs:maxInclusive value="20" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="anneeNaissance">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{4}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="code">
    <xs:restriction base="xs:ID">
      <xs:pattern value="I\d{4}" />
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="bulletins">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="cours" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="nom" type="xs:string" use="required" />
            <xs:attribute name="code" type="code" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element name="bloc" minOccurs="3" maxOccurs="3">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="etudiant" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="note" maxOccurs="unbounded" minOccurs="0">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="note">
                            <xs:attribute name="cours" type="xs:IDREF" use="required"/>
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute name="nom" type="xs:string" use="required" />
                  <xs:attribute name="anneeNaissance" type="anneeNaissance" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="numero" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="1"/>
                  <xs:enumeration value="2"/>
                  <xs:enumeration value="3"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Nom et prénom :
Login examen :

A la fin de l'examen, dans votre répertoire bulletins, les trois fichiers suivants devront être présents : bulletins.dtd, bulletins.xq et bulletins.xsl.

DTD [3 pts]

On vous demande d'écrire la DTD bulletins.dtd qui permet de valider les mêmes documents que bulletins.xsd (tout en étant plus permissive évidemment)

XQuery [2 pts]

Dans un fichier bulletins.xq, écrivez la requête qui donne pour chaque cours la meilleur note obtenue par un étudiant pour ce cours (avec l'aide de BaseX). Le résultat devrait ressembler à cela :

```
<notesMax>
  <cours nom="Structure de donnees">17</cours>
  <cours nom="Bases de donnees">19</cours>
  <cours nom="Mobile">19</cours>
</notesMax>
```

XSL [3pts]

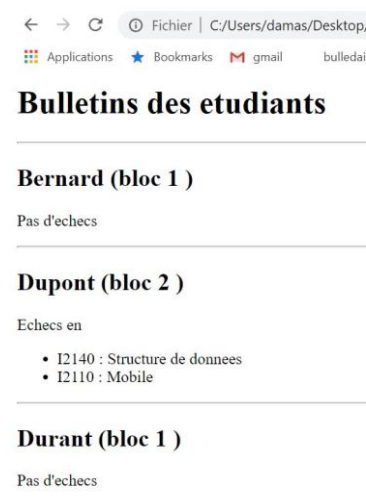
On vous demande d'écrire un fichier bulletins.xsl qui transforme un fichier xml valide par rapport à bulletins.xsd en html.

Le fichier généré affiche tous les étudiants triés par ordre alphabétique avec leur numéro de bloc et leurs échecs (note strictement inférieure à 10). Pour un échec, on affiche le code et le nom du cours (séparé par un « : »). Si un étudiant n'a pas d'échecs, il faut afficher « Pas d'échecs ».

Remarque : Si l'on veut mettre un "<" ou un ">" dans une expression xpath en xsl, il faut les coder avec "<" (pour "<") ou ">" pour ">") car les symboles ">" et "<" sont réservés pour les balises. (voir pg 19 du syllabus)

Exemple: l'expression xpath "//bloc[@numero<3]" renvoie les deux premiers blocs

Ci-dessous, vous trouverez la capture d'écran ainsi que le fichier source d'un bulletins.html obtenu à partir d'un fichier xml contenant les informations suivantes : le bloc 1 contient les étudiants Bernard et Durant n'ayant pas d'échecs, le bloc 2 contient l'étudiant Dupont ayant deux échecs en Mobile et Structures de données, et le bloc 3 ne contient pas d'étudiants.



← → ↻ ⓘ Fichier | C:/Users/damas/Desktop, Applications Bookmarks gmail bulledai

Bulletins des etudiants

Bernard (bloc 1)

Pas d'echecs

Dupont (bloc 2)

Echecs en

- I2140 : Structure de donnees
- I2110 : Mobile

Durant (bloc 1)

Pas d'echecs

Nom et prénom :

Login examen :

```
<html>
  <head>
    <title>Bulletins</title>
  </head>
  <body>
    <h1>Bulletins des etudiants</h1>
    <hr/>
    <h2>Bernard (bloc 1)</h2>
    <p>Pas d'echecs</p>
    <hr/>
    <h2>Dupont (bloc 2)</h2>
    <p>Echecs en
      <ul>
        <li>I2140 : Structure de donnees</li>
        <li>I2110 : Mobile</li>
      </ul>
    </p>
    <hr/>
    <h2>Durant (bloc 1)</h2>
    <p>Pas d'echecs</p>
    <hr/>
  </body>
</html>
```

(Remarque : lors de la génération, il est normal que votre fichier html ne soit pas indenté et qu'il y ait 2 différences avec le fichier ci-dessus : les balises <hr> et une balise <META>)

Nom et prénom :
Login examen :

b) Graphe [5 pts]

Dans le répertoire graphe, vous trouverez un document XML valide intitulé `vols.xml`. Ce fichier contient des informations à propos de vols aériens.

Ce fichier contient un certain nombre d'aéroports (élément `airport`) identifié par un code appelé code iata. Pour chaque aéroport, on a une liste de vol partant de cet aéroport (élément `flight`). Pour chaque vol, on a le code iata de l'aéroport de destination ainsi que la compagnie (attribut `airline`).

Ce fichier peut être vu comme un graphe dirigé. Les sommets de ce graphe sont les aéroports. Il y a un arc entre deux aéroports si un vol les relie.

Dans le répertoire Graphe, on vous fournit un squelette de code : la classe `Graph.java` et la classe `Vol.java`. L'implémentation de la classe `Graph` est basée sur une liste d'arcs. Cette classe va retenir dans la liste `vols` l'ensemble des vols du document xml.

L'objectif de cette question est de compléter la classe `Graphe.java`.

Parseur SAX [2 pts]

La classe `Graphe.java` contient une classe interne `SAXHandler`. Ecrivez cette classe interne qui permettra de remplir la structure de données `vols`, une liste de "Vol" qui contiendra l'ensemble des vols du document xml.

Comme `SAXHandler` est une classe interne non statique, elle peut accéder et modifier l'attribut `vols` de l'objet qui l'a créé.

BFS [3pts]

Implémentez la méthode `bfs()` de la classe `Graphe` qui prend le code iata d'un aéroport de départ en paramètre. Cette méthode affiche à la sortie standard les codes iata des différents aéroports qu'il peut atteindre dans l'ordre d'un parcours en largeur (BFS) depuis l'aéroport de départ.

Par exemple, une des sorties possibles de la méthode `g.bfs("JFK")` est :

```
JFK BCN CDG DFW FCO LAX LHR MAD ORD ATL FRA DEN DXB PEK AMS MUC IST LGW STN DME  
EWR IAH
```

Nom et prénom :

Login examen :

c) Récursion [1 pt]

Dans le répertoire `recursion`, vous trouverez la classe `Tree.java`.

contains [1 pt]

Dans la classe `Tree`, implémentez la méthode récursive `contains(int x)` qui renvoie vrai si l'arbre contient l'entier dans un de ses nœuds.

Une classe `Main` est fournie. La sortie attendue est :

```
true  
false
```

d) Huffman [2pts]

Dans le package `Huffman`, vous trouverez un projet presque complet qui implémente la méthode de Huffman pour compresser et décompresser un fichier. Implémentez la seule méthode manquante : `buildCode(Node root)`. Cette méthode construit le code de chaque lettre à partir de l'arbre passé en paramètre.

Nom et prénom :
Login examen :

e) Football [3 pts]

Dans le répertoire football, nous fournissons un squelette de code qui a pour but de gérer un championnat de football en aller-retour. Chaque équipe du championnat est identifiée par son nom et va jouer deux matchs contre chaque équipe: une fois à domicile et une fois chez l'équipe visiteuse. Par exemple, on aura le match IPL (domicile)-ECAM (visiteur) et le match ECAM(domicile)-IPL(visiteur) mais on ne peut pas avoir deux matchs avec la même équipe domicile et la même équipe visiteuse.

Les points du championnat sont calculés comme suit :

- Si un match se termine par une victoire, l'équipe victorieuse obtient 3 points et l'équipe vaincue n'obtient pas de points.
- Si un match se termine par une égalité, les deux équipes obtiennent 1 point.

L'objectif de cette question est de compléter les deux méthodes manquantes de la classe Championnat : `encoderMatch()` et `afficherClassement()`. Vous devez garantir une efficacité maximale pour ces méthodes.

Vous pouvez ajouter/modifier des attributs dans la classe Championnat et vous pouvez également compléter le constructeur de cette classe. Vous pouvez également rajouter des méthodes dans Equipe et Match si nécessaire.

Dans les cadres ci-dessous, donnez les complexités de vos deux méthodes en fonction de n (nombre d'équipes) et m (nombre de matchs) :

`encoderMatch()`:

`afficherClassement()`:

Une méthode main est fournie. L'output attendu est le suivant :

```
1. ipl 6
2. ucl 4
3. ecam 1
4. ephec 0
```

```
Exception in thread "main" java.lang.RuntimeException: match déjà joué
    at Championnat.encoderMatch(Championnat.java:35)
```


Nom et prénom :
Login examen :

2. Question sur papier [1 pt]

Soit une structure Union-Find représenté par le tableau à deux dimensions suivant :

# du sous-arbre	1	1	1	5	3	4	2
Indice parent	6	4	4	-1	5	3	-1
	0	1	2	3	4	5	6

A quoi ressemblerait ce tableau après l'appel de la méthode `find(2)` ?