TD 10
CREATION D'UNE BASE DE DONNEES AVEC JDBC

Buts poursuivis par ce TP

- Utiliser l'API Java JDBC.
- Mettre en œuvre les expressions régulières.

Sujet

Il s'agit créer une base de donnée constituée à partir d'une liste d'ouvrage techniques et scientifiques (*Ouvrages.txt*) récupérée sur différents sites d'éditeurs. Le fichier *Ouvrages.txt* se présente sous la forme donnée en annexe 1.

- La première ligne (entête) du fichier contient le nom des champs. Les lignes suivantes contiennent les données relatives aux champs.
- Chaque champ dans l'entête et les données est séparé par un « ; ». Un champ vide est encadré par deux « ; ».
- La base de données à créer portera le nom de « `OUVRAGES` » et utilisera comme clé un entier. Les champs (colonnes) seront identifiés (nommés) à l'aide des informations fournies à la ligne 1 du fichier *`Ouvrages.txt`*.
- La classe métier (celle gérant la base de donnée) portera le nom de `BddOuvrages` est devra être suffisamment souple pour s'adapter à un nombre de champs (colonnes) quelconques.

Description des méthodes

Constructor Detail

BddOuvrages

```
public BddOuvrages(java.lang.String path,
                   java.lang.String name,
                   java.lang.String user,
                   java.lang.String password)
            throws java.lang.ClassNotFoundException,
                   java.sql.SQLException
```

Charge le pilote H2

Se connecte à la base de donnée et mémorise la référence de la connexion dans un attribut `con` de type `Connection`.

Création d'un objet `Statement` et mémorisation de sa valeur dans l'attribut `stmt` de type `Statement`.

Parameters:

`path` - : emplacement de la base de données

`name` - : nom de la base de données

`user` - : nom de l'utilisateur de la BDD

`password` - : mot de passe d'accès à la BDD

Throws:

`java.lang.ClassNotFoundException, java.sql.SQLException`

Method Detail

createTable

```
public void createTable(java.lang.String table,
                        java.lang.String[] field)
                 throws java.sql.SQLException
```

Crée une table dans la BDD courante. Détruit cette table si elle existe.

La table est crée avec comme clé primaire un entier et des champs de type `VARCHAR(255)`

Parameters:

`table` - : nom de la table

`field` - : tableau contenant le nom des champs

Throws:

`java.sql.SQLException`

addRecord

```
public void addRecord(int num, java.lang.String[] records)
               throws java.sql.SQLException
```

Ajoute un enregistrement dans la BDD courante

Parameters:
`num` - : la clé de l'enregistrement
`records` - : tableau contenant l'ensemble des colonnes.
Throws:
`java.sql.SQLException`

---

getBdd

```
public java.lang.String getBdd(java.lang.String table)
                        throws java.sql.SQLException
```

Obtient le contenu de la BDD courante
Parameters:
`table` - : nom de la table désirée
Returns:
: chaine de caractères contenant le contenu de la BDD
Throws:
`java.sql.SQLException`

---

close

```
public void close()
            throws java.sql.SQLException
```

Ferme la connexion à la base de données
Throws:
`java.sql.SQLException`

Travail demandé

Nota : on pourra s'inspirer d'un exemple de code donné annexe 4.

1. Importer le fichier H2.jar dans le répertoire `bin` du projet ou ajouter la bibliothèque H2.jar au chemin de compilation.
   Définir la classe métier `BddOuvrages`. Définir le constructeur (voir documentation de la classe).
   Ecrire un programme de test : `TestUnitaire`.java instanciant la classe `BddOuvrages`.
   Nom de la base de données : `OUVRAGES`, nom d'utilisateur `sa`, pas de mot de passe.
   Vérifier la création de la base de données à l'aide de la console H2.
2. Définir la méthode `createTable` (voir documentation de la classe).
   Programme de test : appeler la méthode `createTable` en lui passant le nom des champs extraits de la première ligne du fichier `ouvrage.txt`.
   Vérifier la création de la base de données à l'aide de la console H2.
3. Définir la méthode `addRecord` (voir documentation de la classe).
   Programme de test : pour chaque ligne du fichier `ouvrage.txt`, appeler la méthode `addRecord` en lui passant dans un tableau la valeur des différents champs extraits de chaque ligne du fichier.
   Nota : le texte à mémoriser dans les différents champs peuvent contenir des caractères « ' ». Ce caractère fait partie du jeu de caractères du langage SQL et sera interprété dans les requêtes comme tel. Il faudra remplacer ce caractère par deux caractères identiques.
   A l'aide de la console H2, effectuer une requête permettant d'afficher le contenu de la base de données `OUVRAGES`
4. En vous basant sur le format de la chaine de caractère retournée par la méthode `getBdd` annexe 2, donner l'algorithme général de cette fonction.
   Commenter le code de cette fonction donné dans l'annexe 3.

ANNEXE 1 : FORMAT DES DONNEES BRUTES (EXTRAIT)

TITRE;AUTEURS;ÉDITEUR;NIVEAU;MOTS_CLES
Réseaux informatiques : conception et optimisation; Malek Rahoual & Patrick Siarry; Technip; élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs; ;
Conception des Systèmes d'Information;M. Bigand, JP Bourey, H. Camus, D. Corbeel;Technip;élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs; ;
Automatisation des processus dans l'espace état. Cours et exercices d'application résolus;P. Borne, Ph. Vanheeghe, E. Duflos;Technip;élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs; ;

ANNEXE 2 : FORMAT DE LA CHAINE DE CARACTERES RETOURNEE PAR LA METHODE getBdd

ID	TITRE	AUTEURS	ÉDITEUR	NIVEAU	MOTS_CLES
1	Réseaux informatiques : conception et optimisation	Malek Rahoual & Patrick Siarry	Technip	élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs	
2	Conception des Systèmes d'Information	M. Bigand, JP Bourey, H. Camus, D. Corbeel	Technip	élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs	
3	Automatisation des processus dans l'espace état. Cours et exercices d'application résolus	P. Borne, Ph. Vanheeghe, E. Duflos	Technip	élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs	
4	Commande vectorielle de la machine asynchrone. Désensibilisation et optimisation par logique floue	B. Robyns, B. François, P. Degobert, JP Hautier	Technip	élèves-ingénieurs, étudiants de masters, ingénieurs Recherche & Développement et chercheurs	
5	Initiation au Grafcet	S.MORENO, E.PEULOT	Casteilla	Lycée Technologique	
6	Le GEMMA	S.MORENO, E.PEULOT	Casteilla	Lycée Technologique, STS, IUT, École d'ingénieurs	
7	Le GRAFCET sa pratique et ses applications	JC BOSSY, P BRARD, P FAUGERE, C MERLAUD	Casteilla	Lycée Technologique, STS, IUT, École d'ingénieurs	
8	Le GRAFCET conception ~ implantation dans les API	S.MORENO, E.PEULOT	Casteilla	Lycée Technologique, STS, IUT, École d'ingénieurs	
9	Systèmes asservis linéaires	JC & P CHAUVEAU	Casteilla	STS, IUT, École d'ingénieurs	
10	Systèmes asservis linéaires et non linéaires exercices & problèmes résolus	JC & P CHAUVEAU	Casteilla	STS, IUT, École d'ingénieurs	
11	Automatique informatique industrielle livre de 1ère & livre de terminale	C ROBINET, A BIANCIOTTO, P BOYE	Delagrave	Lycée Technologique	
12	Automatique informatique industrielle	C MERLAU, J PERRIN, JP TRICHARD	Dunod	Lycée Technologique,STS, IUT, École d'ingénieurs	
13	Les automatisme programmables	D & N BOUTEILLE	Cepadues	Lycée Technologique STS, IUT, École d'ingénieurs	
14	Automatique Informatique Industrielle	F.BENIELLI, G.CERATO	Foucher	Lycée Technologique Lycée Professionnel	
15	Les points clés de l'automatisme	M LABIT, D VIVIER	Foucher	Lycée Technologique,STS, IUT, École d'ingénieurs	
16	Maintenance des systèmes mécaniques automatisés	A GEORJON, H BERTHOMIEU	Hachette	Lycée Professionel seconde et terminale	
17	Automatique et informatique industrielle	Y MONPEURT, M OUDJEDI	Hachette	Lycée Technologique	
18	Automatique informatique industrielle, livre de 1ère & terminale	H.NEY	Nathan	Lycée Technologique	
19	ÉTAPES : Génie mécanique Automatique industrielle Mémento	JM.BLEUX , JL FANCHON	Nathan	Lycée Technologique	
20	ÉTAPES : Automatique industrielle Mémento	JM.BLEUX , JL FANCHON	Nathan	Lycée Technologique	
21	ÉTAPES : Pneumatique industrielle Mémento	JM.BLEUX	Nathan	Lycée Technologique	
22	Estimation Prédiction	E DUFLOS P VANHEEGHE	Technip	STS, IUT, École d'ingénieurs	
23	Identification et commande numérique des procédés industriels	R BEN ADBENNOUR, P BORNE,M KSOURI, F M'SAHLI	Technip	STS, IUT, École d'ingénieurs	
24	Commande en temps réel	D TSCHIRHART	Dunod	STS, IUT, École d'ingénieurs, Recherche	
25	La commande par calculateur application aux procédés industriels	M.KSOURI, P.BORNE	Technip	STS, IUT, École d'ingénieurs	
26	Asservissements et régulations continus analyse et synthèse	E.BOILLOT	Technip	STS, IUT, École d'ingénieurs	
27	Régulation industrielle problèmes résolus	M.KSOURI, P.BORNE	Technip	STS, IUT, École d'ingénieurs	
28	Commande et optimisation des processus	P.BORNE, G.DAUPHIN-TANGUY, J-P.RICHARD, F.ROTELLA, I.ZAMBETTAKIS	Technip	STS, IUT, École d'ingénieurs	
29	Modélisation et identification des processus tome 1 & 2	P.BORNE, G.DAUPHIN-TANGUY, J-P.RICHARD, F.ROTELLA, I.ZAMBETTAKIS	Technip	STS, IUT, École d'ingénieurs	
30	Analyse et régulation des processus industriels tome 1 : régulation continue tome 2 : régulation numérique	P.BORNE, G.DAUPHIN-TANGUY, J-P.RICHARD, F.ROTELLA I.ZAMBETTAKIS	Technip	STS, IUT, École d'ingénieurs	
31	Modélisation et commande de la machine asynchrone	J-P.CARON, J-P.HAUTIER	Technip	STS, IUT, École d'ingénieurs	
32	Réalisation réduction et commande des systèmes linéaires	A.RACHID D.MEHDI	Technip	STS, IUT, École d'ingénieurs	
33	Le langage JAVA	T. Leduc, D Leduc	Technip	STS,IUT,Ecole d'ingénieurs	
34	Les logiciels de gestion hautement intégrés	J-M. Tysebaert	Technip	STS,IUT,Ecole d'ingénieurs	
35	MATLAB-Simulink-Stateflow, avec exercices d'automatique résolus	M.Rivoire, J-P. Ferrier	Technip	STS,IUT,Ecole d'ingénieurs	
36	Théorie et traitement du signal	M BENIDIR	Technip	STS-IUT-IUP-Ecoles d'ingénieurs - 2e Cycle DEA	
37	Automatique des systèmes échantillonnés	Ph. Vanheeghe, C. Sueur, P. Borne	Technip	STS,IUT,Ecole d'ingénieurs	

ANNEXE 3 : METHODE RETOURNANT LE CONTENU FORMATE DE LA BASE DE DONNEES

```
public String getBdd(String table) throws SQLException
{
    resultSet = stmt.executeQuery("SELECT * FROM " + table);
    ResultSetMetaData rsmd = resultSet.getMetaData();
    int nbCols = rsmd.getColumnCount();
    int[] rawSize = new int[nbCols];
    // On démarre à la colonne 2, la colonne 1 contenant un entier
    for (int i = 2; i <= nbCols; i++)
    {
      rawSize[i-1] = rsmd.getColumnName(i).length();
    }
    boolean encore = resultSet.next();
    while (encore)
    {
      for (int i = 2; i <= nbCols; i++)
      {
        int size = resultSet.getString(i).length();
        if (rawSize[i-1] < size) rawSize[i-1] = size;
      }
      encore = resultSet.next();
    }
```

```
    resultSet = stmt.executeQuery("SELECT * FROM " + table);
    rsmd = resultSet.getMetaData();
    encore = resultSet.next();


    String s="";
    s += String.format("%3s |", rsmd.getColumnName(1));
    for (int i=2; i <= rawSize.length; i++)
    {
      s += String.format("%-" + ((rawSize[i-1]==0) ? 1 : (rawSize[i-1])) + "s | ", rsmd.getColumnName(i));
    }
    s += "\r\n";

    String line = "";
    for (int i=0; i < s.length(); i++)
    {
      line += "-";
    }
    s += line + "\r\n";

    while (encore)
    {
      s += String.format("%3s |", resultSet.getString(1));
      for (int i = 2; i <= nbCols; i++)
      {
        s += String.format("%-" + ((rawSize[i-1]==0) ? 1 : (rawSize[i-1])) + "s | ", resultSet.getString(i));
      }
      encore = resultSet.next();
      s += "\r\n";
    }
    return s;
  }
```

ANNEXE 4 : EXEMPLE DE CREATION D'UNE BASE DE DONNEES INVENTAIRE

```
import java.sql.*;

public class TestJdbc1
{
  static Statement stmt = null;

  private static void affiche(String message)
  {
    System.out.println(message);
  }
```

```
  private static void afficheBase()
  {
    affiche("Parcours des données");
    try
    {
      ResultSet resultSet = stmt.executeQuery("SELECT * FROM " + "Inventaire");
      ResultSetMetaData rsmd = resultSet.getMetaData();
      int nbCols = rsmd.getColumnCount();
      boolean encore = resultSet.next();

      while (encore)
      {
        for (int i = 1; i <= nbCols; i++)
          System.out.print(resultSet.getString(i) + "\t");
        System.out.println();
        encore = resultSet.next();
      }
    }
    catch (SQLException e) { arret(e.getMessage()); }
  }

  private static void arret(String message)
  {
    System.err.println(message);
    if (message.matches("\\*{3}.*"))  // recherche de ***
      System.exit(99);
  }

  public static void main(java.lang.String[] args)
  {
    Connection con = null;
    ResultSet  resultSet = null;
    // Chargement du pilote H2
    try { Class.forName("org.h2.Driver"); }
    // Chargement du pilote MySQL
    // try { Class.forName("org.gjt.mm.mysql.Driver").newInstance(); }
    catch (Exception e)
    {
      e.printStackTrace();
      arret("*** Impossible de charger le pilote jdbc");
    }
    affiche("Connection à la base de données H2");
    try { con = DriverManager.getConnection("jdbc:h2:d:/bddtest", "sa", ""); }
//  try { con = DriverManager.getConnection("jdbc:h2:~/bddtest", "sa", ""); }
//  try { con = DriverManager.getConnection("jdbc:h2:tcp://localhost/~/bddtest", "sa", ""); }
//  try { con = DriverManager.getConnection("jdbc:h2:tcp://ASUS/~/bddtest", "sa", ""); }
```

```
// Connexion à une base MySQL
//  try { con = DriverManager.getConnection("jdbc:mysql://localhost/testMsql", "root", "");  }
    catch (SQLException e)
    {
      e.printStackTrace();
      arret("*** Connection à la base de données impossible");
    }

    try
    {
      stmt = con.createStatement();

      // Suppression de la base Inventaire si elle existe (phase de test)
      stmt.executeUpdate("DROP TABLE IF EXISTS Inventaire;");
      // Création de la base Inventaire : clé : entier, données Déignation : chaine, ... Quantité : entier
      stmt.executeUpdate("CREATE TABLE Inventaire(ID INT PRIMARY KEY, Designation VARCHAR(255),
                          Caracteristiques VARCHAR(255), Quantite int);");
      // Insertion des enregistrements dans la table Inventaire
      affiche("Creation des enregistrements");
      String requete1 = "INSERT INTO Inventaire VALUES (1, 'Vis', '10',  80)";
      String requete2 = "INSERT INTO Inventaire VALUES (2, 'Vis', '20',  200)";
      String requete3 = "INSERT INTO Inventaire VALUES (3, 'Ecrous', '100', 50)";
      stmt.executeUpdate(requete1);
      stmt.executeUpdate(requete2);
      stmt.executeUpdate(requete3);
    }
    catch (SQLException e)
    {
      e.printStackTrace();
      arret("*** Requète incorecte");
    }

    // Création et exécution de requêtes
    affiche("Création et exécution de requêtes");

    try
    {
      // Préparation des commandes sql
      // Sélection de la base complète
      String requete4 = "SELECT * FROM Inventaire";
      // Sélection d'un enregistrement particulier
      String requete5 = "SELECT * FROM Inventaire WHERE Designation = 'Vis' AND Quantite < 100";

      // Création d'un objet permettant l'envoi de commandes vers la base de données
      Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);
      // Exécution de la requète 1
      resultSet = stmt.executeQuery(requete4);
```

```
      // Affichage des résultats
      afficheBase();
      // Exécution de la requète 2
      resultSet = stmt.executeQuery(requete5);
      // Lecture de l'enregistrement correspondant
      resultSet.next();
      // On désire incrémenter la donnée mémorisée dans le champ quantité
      // Lecture de la donnée. Elle est située colonne 4 (les colonnes sont numérotées à partir de 1
      int q = resultSet.getInt(4) + 1;
      // Mise à jour de l'enregistrement (la ligne)
      resultSet.updateInt("Quantite", q);
      // ou en désignant le numéro de la colonne
      //resultSet.updateInt(4, 11);
      // Mettre à jour l'enregistrement (provoque le positionnement sur l'enregistrement suivant)
      resultSet.updateRow();
      // Afficher le résultat
      // Reculer
      resultSet.previous();
      afficheBase();
      // Clore
      resultSet.close();
      con.close();
    }
    catch (SQLException e)
    {
      e.printStackTrace();
      arret("*** Anomalie lors de l'exécution de la requête");
    }
    arret("Bye");
  }
}
```