TP n°5 : Java DataBase Connector (JDBC)

On souhaite réaliser une application qui permette de consulter et de modifier des informations contenues dans une base de données. La base de données « inventaire » comporte une table permettant d'enregistrer des noms d'articles ainsi que les quantités disponibles pour chaque article.

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra
☐	id	int(11)			Non		auto_increment
☐	nom	tinytext	latin1_general_cs		Non		
☐	quantite	int(11)			Oui	*NULL*	

id	nom	quantite
1	Perceuse	3
2	Tournevis électrique	85
3	Coupe bordure	38
4	Tondeuse à gazon	18
5	Nettoyeur HP	1
6	Scie sauteuse	7
7	Remorque 700kg	5
8	Groupe electrogène	2
9	Raton laveur	24
10	Aspirateur	58
11	Oscilloscope	12
12	Planche à voile	72

Votre application doit permettre :

- d'afficher le contenu de la table ;
- de retirer (ou déposer) un article ; c'est-à-dire décrémenter (ou incrémenter) la quantité associée à un article.

Les informations nécessaires pour établir la connexion à la base de données sont les suivantes :

- url : jdbc:mysql://192.168.2.5/inventaire
- user : guest
- password : guest
- driver : com.mysql.jdbc.Driver

Pour installer le driver, vous devez décompacter le fichier « mysql-connector-javaX.X.XX-bin.jar » dans votre répertoire de travail : votre classe utilisant le driver jdbc doit se trouver au niveau des répertoires « org » et « com » obtenus après la décompression de l'archive .jar. Vous devez importer le package (**import java.sql.* ;**) pour l'utiliser.

1. Une classe représentant les éléments de la BDD

Vous allez tout d'abord créer une simple classe ElementBDD représentant un champ de votre BDD. Il y aura 2 attributs, une chaine *nom* et un entier *quantite*. Vous écrirez un constructeur permettant de créer un objet de cette classe avec deux paramètres permettant d'initialiser les attributs. Vous écrirez ensuite 2 méthodes « getter » permettant de retourner le nom et la quantité.

2. Une classe permettant de gérer la BDD

Vous devez ensuite écrire une classe *GestionBDD* permettant de vous connecter à la BDD, de lister, d'ajouter, et de déposer des articles. Pour cela, vous utiliserez en attribut de votre classe un objet de type ArrayList comportant des objets de type ElementBDD. Cet attribut permet de gérer dynamiquement des listes d'objet. La déclaration s'effectue de la façon suivante :

```
private ArrayList<ElementBDD> list;
```

Écrivez un constructeur à votre classe. Pour initialiser votre liste, vous devez faire :

```
this.list = new ArrayList<ElementBDD>();
```

Vous utiliserez ensuite les méthodes suivantes, en prenant soin de les adapter à votre code :

- Méthode permettant de lister les articles de la BDD

```
public void listerArticles() {
  Connection con = null;
  Statement st = null;
  ResultSet rs = null;
  try {
     Class.forName("mettre ici le nom du driver").newInstance();
     con = DriverManager.getConnection("mettre ici l'url de la BDD",
"ici le user", "ici le password");
     st = con.createStatement();
     rs = st.executeQuery("SELECT id, nom, quantite FROM articles");
     // remise à 0 de la liste - utile pour les mises à jour
     list.clear();
     // Stocker les enregistrements dans la liste
     while(rs.next()) {
         int id = rs.getInt(1);
         String nom = new String(rs.getString(2));
         int quantite = rs.getInt(3);
         list.add(new ElementBDD(nom, quantite));  // ajout
         }
     }
     catch (Exception e) {
```

```
            System.err.println("Exception: " + e.getMessage());
        }
        finally {
            try {
             if(rs != null) rs.close();
             if(st != null) st.close();
             if(con != null) con.close();
            }
            catch (SQLException e) {
            }
    }
}
```

- Méthodes pour retirer et ajouter un article

```
public void retirerArticle(String nomArticle) {
      mettreAjourQuantite(nomArticle,-1);
}

public void deposerArticle(String nomArticle) {
      mettreAjourQuantite(nomArticle,1);
}
```

- Les méthodes **retirerArticle** et **deposerArticle** utilisent la méthode **mettreAJourQuantite**

```
public void  mettreAjourQuantite(String nomArticle,int value) {
   Connection con = null;
   Statement st = null;
   ResultSet rs = null;
   try {
     Class.forName("mettre ici le nom du driver").newInstance();
     con = DriverManager.getConnection("mettre ici l'url de la BDD",
"ici le user", "ici le password");
     st = con.createStatement(
                ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
     rs = st.executeQuery("SELECT id, nom, quantite FROM articles");
     // modifier la quantité pour l'article nomArticle
     while(rs.next()) {
         int id = rs.getInt(1);
         String nom = rs.getString(2);
         int quantite = rs.getInt(3);
         if ( nom.equals(nomArticle) ) {
          if ( (quantite == 0) && (value == -1) ) {
              break;
          }
          rs.updateInt(3, (quantite+value));
          rs.updateRow();
```

```
            break;
          }
        }
    } catch (Exception e) {
       System.err.println("Exception: " + e.getMessage());
    }
    finally {
         try {
           if (rs != null) rs.close();
           if (st != null) st.close();
           if (con != null) con.close();
         } catch (SQLException e) {
         }
    }
}
```

Incluez ces méthodes dans votre code et testez leur bon fonctionnement avec la BDD en faisant afficher dans les console les enregistrements de la BDD, puis en testant de retirer et d'ajouter des quantités.

3. Une classe pour l'interface graphique

On a l'habitude dans tout code gérant une interface graphique de bien séparer l'affichage des traitements. L'intérêt est de pouvoir facilement changer l'un ou l'autre indépendemment. Ainsi, si la méthode de connexion à la BDD est modifiée (PostGreSQL, données dans des fichiers XML…), on change seulement la classe s'occupant de cette partie. La classe d'affichage reste la même. Dans tout projet, cette méthode de développement est essentielle pour découper le travail et assurer une flexibilité au code.

Ecrire une classe *InterfaceBDD* permettant d'afficher une fenêtre graphique.

Pour vous aider, voici une méthode utile pour créer cette interface. Certains objets de cette méthode seront utiliseés dans d'autres méthodes de la classe : vous devez donc prendre soin de bien les déclarer en attribut de votre classe.

```
public void creerInterface() {
  JLabel titre = new JLabel("Articles - quantité");
  getContentPane().add(titre, BorderLayout.NORTH);
  this.list = new JList();
  getContentPane().add(list, BorderLayout.CENTER);
  JPanel p = new JPanel();
  this.but_dep = new JButton("Dépot");
  p.add(but_dep);
  but_dep.addActionListener(this);
  this.but_ret = new JButton("Retrait");
```

```
  p.add(but_ret);
  but_ret.addActionListener(this);
  this.but_maj = new JButton("Mise à jour");
  p.add(but_maj);
  but_maj.addActionListener(this);
  this.but_esc = new JButton("Quitter");
  p.add(but_esc);
  but_esc.addActionListener(this);
  getContentPane().add(p, BorderLayout.SOUTH);
  this.pack();
}
```

Vous remarquez que nous allons afficher les élements de notre BDD dans un objet de type JList. Pour afficher les éléments de notre ArrayList de la classe GestionBDD, voici la procédure à suivre :

- Créez un attribut de type GestionBDD dans votre classe InterfaceBDD.
- Implémentez une méthode getter sur votre ArrayList dans la classe GestionBDD.
- Récupérez la liste au moyen du getter dans le constructeur (ou une méthode) de la classe InterfaceBDD.
- Ajoutez cette liste à votre JList grâce à l'instruction :

```
maJliste.setListData(monArrayListe.toArray());
```

Affichez alors votre interface. Que passe t-il ? Pourquoi avez-vous les adresses d'affichées plutôt que le nom des éléments de la BDD ?

4. Modifiez l'affichage d'un objet de type ElementBDD

Ecrivez une méthode `public String toString()`dans la classe ElementBDD permettant d'afficher le nom et la quantité d'un élément. Retester alors l'affichage dans votre interface graphique. Magique, non ?

Surcharger cette méthode permet de redéfinir l'affichage d'un objet. Ainsi, si vous faites un `System.out.println` sur un objet de type ElementBDD, vous aurez le formattage défini dans la méthode `toString` ().

5. Gérez les boutons

Terminez ensuite votre interface en gérant le clic sur les boutons et en leur associant leur bon comportement. Attention au rafraichissement de votre JList !