Annexe TP2

Ce document fournit une liste de fonctions, et un ensemble d'informations qui pourront vous aider à réaliser le TP2.

Les tables (jTable)

Il est fortement recommander d'utiliser les tables pour afficher les différentes informations relatives à ce TP (les livres, les clients, et les commandes). Dans Netbeans (swing), cela revient à utiliser jTable. Voici une liste des fonctions liées aux tables *jTable*.

- Pour créer une table: à partir de la palette choisir Table (la méthode la plus simple). Si on veut la créer en utilisant le code source (manuellement):

```
String data[][]={ {"101","Amit","670000"},
        {"102","Jai","780000"},
        {"101","Sachin","700000"}};  // les lignes
String column[]={"ID","NAME","SALARY"}; // les titres des colonnes
JTable jt=new JTable(data,column);
jt.setBounds(30,40,200,300);  // la postion et la taille de la table
JScrollPane sp=new JScrollPane(jt);
f.add(sp);          // f est une fenêtre (jframe)
f.setSize(300,400);
f.setVisible(true);   // afficher la fenetre
```

  Le code source ci-dessus permet de créer une table, dont les titres des colonnes sont ID, $NAME$, ET $SALARY$, et les lignes sont définis dans data [][].

- Pour définir les titres des colonnes et leur type: Properties -> Model -> Table Settings.
- Insérer des éléments par défaut: Properties -> Model -> Default values
- Changer la taille de la colonne (clic droit sur la table -> table content -> column -> Min. width)
- Récupérer l'indice de la ligne sélectionnée:

```
int rowIndex = jTable.getSelectedRow();
```

- Supprimer un élément (ligne) de la table :

```
import javax.swing.table.DefaultTableModel;
DefaultTableModel model =  (DefaultTableModel) jTable1.getModel();
// Il faut d'abord  définir le modèle du tableau
model.removeRow(rowIndex); // supprimer la ligne dont l'indice est rowIndex.
```

- Insérer une nouvelle ligne dans la table:

```
String line [] = {"1", "auteur", "Titre", "date"}; // une liste de chaine de
// caractère à insérer dans la table
model.addRow(line) // insérer la liste line dans une nouvelle ligne de la table.
```

- Effectuer une recherche dans la table (filtre utilisant jTextField): dans la fonction correspondant à l'événement de changement de texte dans la zone de saisie (à déterminer), utiliser la fonction suivante:

```
sorter.setRowFilter(....) // sorter est un objet de la classe TableRowSorter
```

- Trier les éléments de la table:

```
setAutoCreateRowSorter(...); // une fonction jTable qui permet de trier la table
selon les titres des colonnes.
```

- Récupérer les éléments d'une ligne sélectionnée:

```
String value = jTable.getValueAt(int rowIndex, int columnIndex).toString();
// retourner la valeur de la ligne rowIndex et la colonne columnIndex
```

Les bases de données (Java DB)

Certaines fonctionnalités dans ce TP nécessitent l'utilisation des base de données. Sous Netbeans, on peut utiliser la base de données intégrée *Java DB*. Dans ce qui suit, une liste de quelques informations utiles relatives à l'utilisation des bases de données.

- Créer une base de donnée:
  Services -> Databases -> clic droit sur JavaDB -> Create Database. Une fenêtre s'affichera dans laquelle il faut introduire le nom de la base de données, ainsi que le nom utilisateur et le mot de passe qui seront utiliser pour se connecter à la base.
- Pour se connecter: cliquer sur le protocole JDBCDerby -> Connect. JDBCDerby est le protocole qui permet de relier le programme java à la base de données.
- Créer une nouvelle table: Dans le sous menu JDBCDerby, cliquer sur Table -> Create table. Par la suite, il faut définir les noms des colonnes, ainsi que leur type.
- Relier un application Java (programme) à la base de données:
  - Créer une nouvelle classe java Application(dans le même projet).

```
import java.sql.*; // import la backage des // bases de données

Connection myConnectObj= null; // pour se connecter à la base
Statement myStatObj = null; //Pour executer les requêts
ResultSet myresObj = null; Pour récupérer les résultats
de l'execution des requêtes

try {
myConnectObj = DriverManager.getConnection ("url_database","user_db","pass_db" );
// se  connecter}
} catch (SQLException ex) {
// error message
}
```

- Utiliser une requête de sélection (**SELECT**).

```
String query = "Select * from databaseName.tableName";
myStatObj = myConnectObj.createStatement();
myresObj = myStatObj.executeQuery(query);
while (myresObj.next()){
int Id = myresObj.getInt ("ID");
}
```

- Utiliser une requête d'insertion (**INSERT**).

```
myStatObj.executeQuery("INSERT INTO Customers " + "VALUES (1001, 'Simpson',
 'Mr.', 'Springfield', 2001)");
```

- Requête de Màj (**UPDATE**):

```
myStatObj.executeQuery("UPDATE warehouses SET name = ? , " + "capacity = ? "
+ "WHERE id = ?"; );
```

- Requête de suppression (**DELETE**):

```
myStatObj.executeQuery("DELETE FROM table_name WHERE condition;");
```

Autres fonctions

- jComboBox1.getSelectedItem().toString(); Renvoi le texte de l'élément sélectionné de la liste (jComboBox).
- jtable.fireTableDataChanged(); pour rafraîchir la table après un changement.
- myresObj.getInt("column_name"); récupérer la valeur de type entier contenue dans la colonne column_name.

- myresObj.getString("column_name"); récupérer la valeur de type String entier contenue dans la colonne column_name.
- myresObj.getDate("column_name"); récupérer la valeur de type date contenue dans la colonne column_name.
- Ajouter un élément à un vecteur:

```
import java.util.Vector;
Vector vec = new Vector();
vec.addElement(myResObj.getObject(i));
// i est l'indice dans la table de la BD
model.addRow(vec);
```