Université du Littoral Côte d'Opale **2016 / 2017**
1ère année Master en Informatique

Base de données avancées
JDBC - TP N°02

Exécuter les commandes SQL suivantes :

```
#
# Structure for the `vol` table :
#
CREATE TABLE vol (
NumVol VARCHAR(8) NOT NULL,
Heure_depart TIME,
Heure_arrive TIME,
Ville_depart VARCHAR(20),
Ville_arrivee VARCHAR(20),
PRIMARY KEY (NumVol)
);
INSERT INTO vol VALUES  ('AF118', '08:30', '10:57','Paris','Athens');
INSERT INTO vol VALUES ('AF212','09:21','14:10','Paris','Moscow');
INSERT INTO vol VALUES ('AF178','12:56','14:15','Paris','London');
INSERT INTO vol VALUES ('TA215','08:00','10:10','Tunis','Paris') ;
INSERT INTO vol VALUES ('OA005','14:20','17:00','Athens','Paris');
INSERT INTO vol VALUES ('SA854','22:00','10:14','Singapore','Athens');
INSERT INTO vol VALUES ('AA111','15:45','21:10','Beijing','Singapore');
INSERT INTO vol VALUES ('AF218','21:12','09:16','Beijing','Paris');
INSERT INTO vol VALUES ('SA012','07:57','11:26','Sydney','Singapore');
INSERT INTO vol VALUES ('AF109','07:39','14:10','Tahiti','Sydney');
INSERT INTO vol VALUES ('AA517','23:57','07:12','Honolulu','Tokyo');
INSERT INTO vol VALUES ('JA014','15:35','19:00','Tokyo','Beijing');
INSERT INTO vol VALUES ('AF002','15:52','00:12','Tokyo','Paris');
INSERT INTO vol VALUES ('JA115','21:26','10:10','Los Angeles','Tokyo');
INSERT INTO vol VALUES ('AA015','20:50','07:00','New York','Lima');
INSERT INTO vol VALUES ('AA515','07:20','12:38','New York','Los Angeles');
INSERT INTO vol VALUES ('AF010','07:53','14:19','Paris New','York');
INSERT INTO vol VALUES ('AF012','07:58','20:10','Paris Los','Angeles');
INSERT INTO vol VALUES ('AA118','07:15','13:10','New York','Paris');
INSERT INTO vol VALUES ('AF001','22:10','12:00','Paris','Tahiti') ;
INSERT INTO vol VALUES ('PA022','10:12','23:55','Lima','Paris');
```

Exercice I :

i. Ecrivez un programme Java qui se connecte à une base de données relationnelle et affiche tous les numéros de vols (issus de la table Vol).

ii. Créer une table avec les données indiquées ci-dessous.

Escales :

Numescale	Ville_escale	Duree_escale
1	Moscou	5
2	Singapour	5
3	Sydney	4
4	Tahiti	4
5	Honolulu	4
6	Los Angeles	5
7	New York	4
8	Londres	3

En utilisant la table Vol, écrivez un programme Java qui propose les vols pour un tour du monde au départ de Paris avec des escales et des durées d'escale prédéfinies dans une table Escales.

iii. Ecrivez un programme Java qui, pour chaque enregistrement de la table Vol, insère un nouvel enregistrement pour le vol retour associé : au Numvol on ajoute le caractère 'R' à la fin, les horaires sont recalculés en considérant un départ 2 heures après l'arrivée (ou sont laissés vides…), la ville de départ et la ville d'arrivée sont échangées. Par exemple, à partir de l'enregistrement :

AF118 08:30 10:57 Paris Athens

On insérera l'enregistrement :

AF118R 12:57 15:24 Athens Paris

Exercice II : Projets

Un employé peut participer à plusieurs projets et un projet peut englober plusieurs employés. (Voir la table ci-dessous) pour créer la table PROJET et la table PARTICIPATION.
Ajoutez un projet dans la table PROJET en tapant directement une requête SQL.

La table PROJET et PARTICIPATION ont été créées sous MYSQL avec :

```
#
# Structure for the `projet` table :
#
create table projet(
codeP varchar(4) constraint pk_projet primary key,
nomP varchar(15) not null
)
#
# Structure for the `participation` table :
#
create table participation(
matr integer constraint fk_emp_part references emp,
codeP varchar(4) constraint fk_projet_part references projet,
constraint pk_part primary key (matr, codeP)
)
```

i. Écrivez en Java une méthode creerEmploye pour créer un nouvel employé dans la base de données.

ii. Appelez cette méthode dans la méthode main et enregistrez ensuite dans cette méthode main le nouvel employé dans le projet que vous avez créé dans la table PROJET.

Rowset :
Vous allez maintenant visualiser et modifier des données de la base sur les employés à l'aide d'une interface graphique.

L'interface utilisateur permet à l'utilisateur de choisir un département et de faire afficher les informations sur les employés de ce département. Le matricule, nom, salaire, date d'embauche et nom du département de chaque employé sont affichés. L'utilisateur peut modifier le salaire des employés affichés ; il ne peut pas modifier les autres informations.

L'interface graphique utilise un RowSet déconnecté pendant tout le temps de la manipulation par l'utilisateur. C'est-à-dire que pendant tout le temps de la manipulation des données par l'utilisateur, les données utilisées proviennent du rowset déconnecté, et pas directement de la base. Pour enregistrer les modifications de l'utilisateur le rowset doit être reconnecté à la base de données. Lors de cette reconnexion, le rowset enregistre uniquement les lignes modifiées par l'utilisateur. Les rowsets sont particulièrement utiles lorsque les données de

la base doivent transiter entre plusieurs couches distantes ; ça n'est pas le cas ici mais, dans de nombreuses applications, l'interface utilisateur est située sur une autre machine que l'ordinateur qui héberge le SGBD.

L'enregistrement des modifications dans la base à la reconnexion du rowset doivent tenir compte des éventuels conflits dus à des modifications effectuées par d'autres utilisateurs pendant la déconnexion du rowset. S'il y a des conflits, le programme demande à l'utilisateur (par une fenêtre de dialogue) s'il veut conserver les données actuellement dans la base ou les écraser par les modifications qu'il a effectuées. Pour tester la gestion des conflits, vous modifierez en direct dans la base, certaines des lignes que vous modifiez en parallèle par l'interface graphique de l'application Java.

Les données de la base sont représentées visuellement par une JTable. Le composant graphique JTable est le composant idéal pour faire afficher des informations "tabulaires" comme le sont les informations enregistrées dans une base relationnelle. Une JTable affiche des données indiquées par un modèle de données, classe qui implémente l'interface TableModel, par exemple, pour savoir ce qu'elle doit afficher dans une case positionnée à une certaine ligne et une certaine colonne, la table appelle la méthode getValueAt(ligne, colonne) du modèle et affiche la valeur renvoyée par cette méthode. Pour le cas de cet exercice, le modèle de données s'appuie sur les données contenues dans un rowset. Vous pouvez étudier le composant JTable dans le tutoriel d'Oracle (le Web comporte de nombreux autres tutoriels sur le composant JTable). Commencez par étudier le code qui vous est donné ci-dessous pour cet exercice

- Classe qui démarre l'application : Main.java
- Interface graphique qui utilise une JTable : GUI.java
- Modèle pour la JTable, qui s'appuie sur un rowset : RowSetTableModel.java
- Une classe utilitaire pour donner les valeurs des propriétés pour la connexion d'un rowset: ConfigConnection.java

Remarque : pour modifier la donnée d'une JTable, l'utilisateur doit commencer par faire un double-clic dans la case de la donnée qu'il veut modifier.

Modifier et compléter le code fournis pour obtenir ce qui vous est demandé. Les parties à modifier sont indiquées par "**********".

Commencez par modifier le code de la classe RowSetTableModel, le modèle de données de la JTable. La classe ConfigConnection contient une méthode pour initialiser un rowset pour la connexion à la base de données. Testez le code que vous avez modifié.

Quelques informations qui pourront vous être utiles :

- Vérifiez avec la javadoc que vous utilisez bien les bonnes méthodes lorsque vous remplacez les "**********" (en particulier pour la méthode setValueAt).
- Si la valeur que vous tapez dans la JTable disparait tout de suite après que vous l'ayez entrée, vous avez sans doute fait une erreur dans la méthode setValueAt (voir information précédente...).
- Si Eclipse vous dit que vous n'avez pas accès à la classe com.sun.rowset.CachedRowSetImpl, un moyen radical de réparer est de supprimer la librairie JRE du projet et de la remettre.