M2106: Bases de données avancées

Feuille de TP n°7

M2106 Bases de données

JDBC

L'objectif de cette feuille de TP est de réaliser un classe qui gère les connexions à votre base de données puis une classe qui gère les mises à jour dans une table `JOUEUR`. La base de données contient une deuxième table `MESSAGE` sera utilisée pour le dernier exercice. Il s'agit des messages échangés entre les joueurs.
Dans l'archive `tp7.zip`, vous trouverez, le script de création de la base de données `joueur.sql`. La colonne `avatar` de la table `JOUEUR` contient des images se qui rend la partie insertion peu lisible. Cette colonne est de type `longblob` qui permet de stocker de type d'informations. Quelques avatars sont proposés dans la répertoire `img`.

```
CREATE TABLE JOUEUR ( numJoueur decimal(6,0), pseudo varchar(10),
  motdepasse varchar(500), sexe char(1), abonne char(1),
  niveau decimal(1,0), avatar longblob,
  PRIMARY KEY (numJoueur), UNIQUE KEY pseudo (pseudo)
);
CREATE TABLE MESSAGE ( idMsg decimal(6,0), dateMsg datetime,
  contenuMsg text, luMsg char(1), idUt1 decimal(6,0),
  idUt2 decimal(6,0), PRIMARY KEY (idMsg),
  FOREIGN KEY (idUt1) REFERENCES JOUEUR(numJoueur),
  FOREIGN KEY (idUt2) REFERENCES JOUEUR(numJoueur)
);
```

Attention ! Les jointures entre ces tables ne sont pas *naturelles* mais se font par des tests entre `numJoueur` et `idUt1` et `idUt2`. Vous allez exécuter le script pour créer dans votre base de données. La table `JOUEUR` contient des images.

Exercice 1 *Structure de l'application*

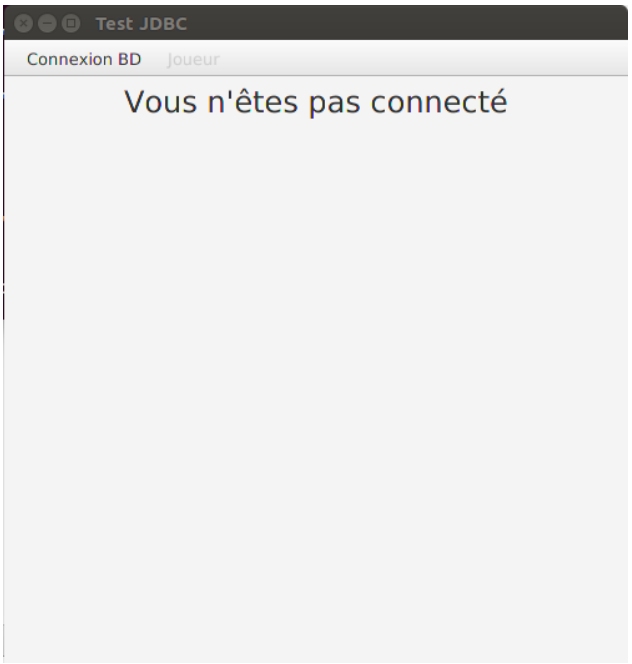
L'application sur laquelle vous allez travailler est une application graphique dont la classe principale est `TestJDBC`. Vous n'aurez à intervenir ni sur les vues ni sur les contrôleurs. Seul les fichiers concernant le modèle (et la base de données) seront à modifier. Il s'agit des fichiers suivants :

- `ConnexionMySQL.java` qui contient le squelette de la classe permettant de charger le driver MySQL et de se connecter à la base de données.
- `JoueurBD.java` qui contient le squelette de la classe permettant d'interagir avec la table `JOUEUR` de votre base de données.

Essayez de compiler et exécuter l'application :

```
javac *.java
java -cp .:/usr/share/java/mysql.jar TestJDBC
```

Normalement vous devriez obtenir la fenêtre suivante



Exercice 2 *La classe* `ConnexionMySQL`

Complétez la classe `ConnexionMySQL` qui contient les attributs `mysql` de type `Connection` et `connecte` qui est un booléen indiquant si la connexion est en cours ou non.

1. Le constructeur de cette classe va simplement charger le driver JDBC pour MySQL.
2. La méthode `connecter` prend en paramètres un nom de serveur, un nom de base de données, un login et le mot de passe de l'utilisateur et va créer la connexion.
3. La méthode `close` va simplement fermer la connexion.

Les exceptions éventuellement levées par ces méthodes seront traitées au niveau de la partie graphique de l'application.
Notez que cette classe contient des reprises des méthodes équivalentes de l'interface `Connection` afin de *simuler* un héritage.
Si votre classe est bien implémentée vous devriez pouvoir vous connecter à la base de données en utilisant l'option *Connexion* du menu *Connexion BD*.

Exercice 3 *La classe* `JoueurBD`

Cette classe va servir à gérer les mises à jour dans la table `JOUEUR` à partir d'un programme JAVA. Cette classe utilise la classe `Joueur` qui permet de représenter les joueurs en mémoire. La photo de l'avatar est stockée sous la forme d'un tableau d'octets (`byte[]`). Pour les interactions avec la base de donnée

— La création d'un blob à partir d'un `byte[]` se fait de la manière suivante :

```
Blob b=laConnexion.createBlob();
b.setBytes(1,j.getAvatar());
ps.setBlob(xxx,b);
```

— La récupération d'un `byte[]` à partir d'un blob lu dans la base de données se fait par la méthode `getBytes` des `ResultSet`.

La classe contient une propriété de type `ConnexionMySQL` qui stocke la connexion avec laquelle on travaille et un objet de type `Statement` qui permet de lancer les ordres SQL. Par ailleurs cette classe devra implémenter les méthodes ci-dessous `Attention!` Testez bien vos méthodes dans la classe `TestJDBC` au fur et à mesure que vous les codez ! La méthode

`rechercherJoueurParNum` doit être implémentée avant les méthodes de mise à jour de la table `JOUEUR`.

- un constructeur prenant en paramètre un objet de la classe `ConnexionMySQL`.
- `int maxNumJoueur()` qui interroge la table `JOUEUR` pour connaître le plus grand numéro de joueur déjà attribué. Si la table est vide la méthode doit retourner 0.
- `Joueur rechercherJoueurParNum(int num)` qui retourne le joueur identifié par num ou jette une exception SQL si aucun joueur ne correspond :
  `throw new SQLException("Joueur non trouvé");`
- `int insererJoueur(Joueur j)` qui insère dans la table `JOUEUR` le joueur passé en paramètres. Cette méthode ne prend pas en compte le numéro du joueur mais elle attribue automatiquement au joueur à l'aide de la méthode précédente.
- `void effacerJoueur(int num)` qui supprime de la table `JOUEUR` le joueur identifié par `num`.
- `void majJoueur(Joueur j)` qui effectue la mise à jour du joueur passé en paramètre. Attention seul le numéro du joueur ne pourra pas être modifié.
- `ArrayList<Joueur> listeDesJoueurs()` qui retourne la liste de tous les joueurs de la base de données.
- String rapportMessage() qui va indiquer pour chaque joueur la liste des messages qu'il a reçus classés par ordre chronologique comme ci-dessous.