

## TP 4

*JDBC*

## 1 – Présentation du TP

*Objet du TP*

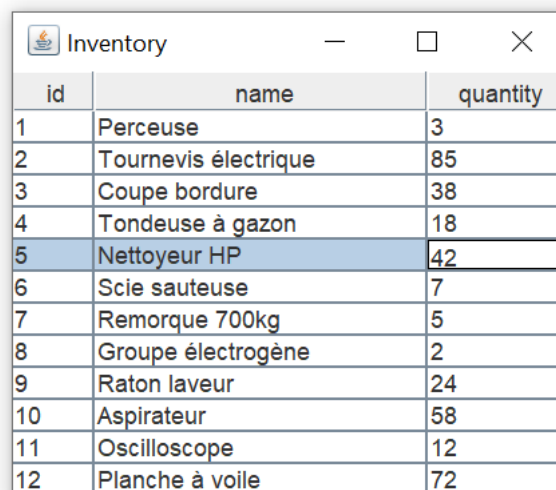
Dans ce TP, nous allons manipuler une base de données MySQL dans un programme Java.

Cette base de données, appelée « inventaire », ne contient qu'une table, appelée « articles », ayant les colonnes suivantes :

Nom	Type	Description
id	Entier	Identifiant du produit
nom	Texte	Nom du produit
quantite	Entier	Nombre de produits en stock

Pour ce faire, nous allons manipuler JDBC, mais aussi les collections et les exceptions Java.

Pour consolider et compléter vos connaissances en Swing, vous allez réaliser une interface graphique qui va permettre de visualiser les données (et les modifier dans une partie optionnelle). Vous vous appuyerez sur un composant appelé JTable, qui permet de réaliser un « mini-tableur », illustré ci-dessous :



id	name	quantity
1	Perceuse	3
2	Tournevis électrique	85
3	Coupe bordure	38
4	Tondeuse à gazon	18
5	Nettoyeur HP	42
6	Scie sauteuse	7
7	Remorque 700kg	5
8	Groupe électrogène	2
9	Raton laveur	24
10	Aspirateur	58
11	Oscilloscope	12
12	Planche à voile	72

## Compétences travaillées dans ce TP

- JDBC
- Collections
- Exceptions
- Swing

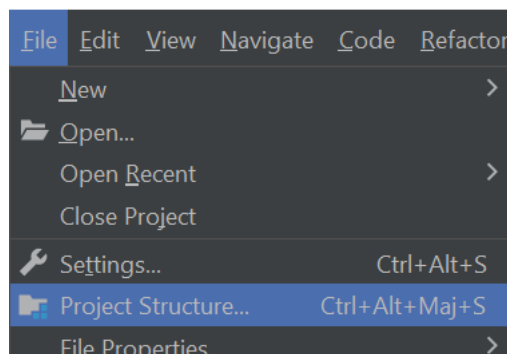
## 2 – Démarrage du TP

Commencez par suivre la même procédure de démarrage que dans les TPs précédents, en nommant votre projet « tp4 ».

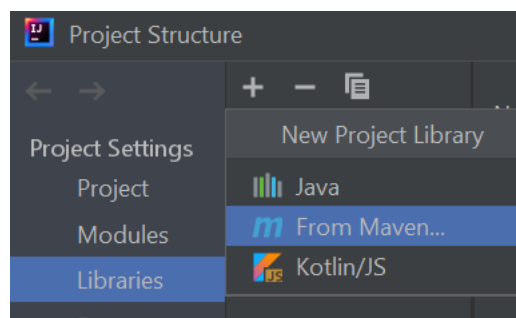
Nous allons à présent configurer le projet afin qu'il charge un *driver* JDBC au démarrage de l'application. Ce *driver* contient un ensemble de classes concrètes qui implémentent les interfaces de JDBC (package java.sql). Ici, nous allons utiliser un driver appelé « Connector/J », permettant de manipuler une base MySQL.

Pour ce faire, suivez les étapes suivantes :

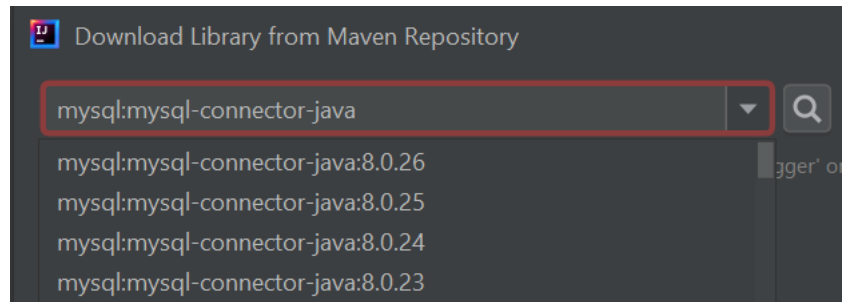
1. Dans le menu, cliquez sur File > Project Structure



2. Dans Project Settings, cliquez sur « Libraries »
3. Cliquez sur + > From Maven



4. Tapez le texte « mysql:mysql-connector-java » et cliquez sur la loupe. Une liste de drivers vous sera proposée.
5. Choisissez le driver ayant la version la plus grande (ici 8.0.26)



6. Cliquez sur OK autant de fois que nécessaire.
7. Ça y est, votre projet est prêt !

En utilisant les fonctionnalités de l'IDE, nous n'avons pas eu à télécharger et à intégrer les fichiers du driver au projet à la main. Cette installation va également nous éviter de charger explicitement le driver en mémoire dans le code du programme (avec `Class.forName()`) : le projet est configuré de telle manière que le driver sera automatiquement chargé au démarrage de l'application.

*Pour les curieux :* A l'étape 3 de la configuration du projet, nous avons cliqué sur « From Maven ». Si vous voulez en savoir plus sur Maven, vous pouvez lire l'encadré ci-dessous.



[Maven](#) est un outil *open source* développé par la fondation Apache permettant de faciliter et d'automatiser certaines tâches de la gestion d'un projet Java. Notamment, Maven permet de gérer les dépendances d'un projet via un système de dépôts. Maven possède un dépôt dit [central](#). Il s'agit d'un dépôt distant contenant de nombreuses bibliothèques, généralement open source, dans lequel nous pouvons effectuer des recherches. Maven télécharge les dépendances du projet dans un dépôt local.

### 3 – Test de la base de données

Les informations de connexion à la base de données « inventaire » sont les suivantes :

- url : `jdbc:mysql://10.10.57.5/inventaire`
- utilisateur : `guest`
- mot de passe : `guest`

Le code suivant permet, en utilisant les informations de connexion ci-dessus, de se connecter à la base de données « inventaire » et d'afficher chaque ligne de la table « articles » dans la console :

```
1 String url = "jdbc:mysql://10.10.57.5/inventaire";
2 String user = "guest";
3 String password = "guest";
4
5 Connection connection = null;
6 Statement statement = null;
7 ResultSet resultSet = null;
8
9 try{
10     connection = DriverManager.getConnection(url, user, password);
11     statement = connection.createStatement();
12     resultSet = statement.executeQuery("SELECT * FROM articles");
13
14     while (resultSet.next()) {
15         int id = resultSet.getInt(1);
16         String name = resultSet.getString(2);
17         int quantity = resultSet.getInt(3);
18         System.out.println(id + " " + name + " " + quantity);
19     }
20 }
21 catch(SQLException e){
22     e.printStackTrace();
23 }
24 finally{
25     try {
26         if (resultSet != null) {
27             resultSet.close();
28         }
29         if (statement != null) {
30             statement.close();
31         }
32         if (connection != null) {
33             connection.close();
34         }
35     }
36     catch (SQLException e) {
37         e.printStackTrace();
38     }
39 }
```

Quelques explications :

- Lignes 10-12 : connexion à la base de données, exécution d'une requête de type SELECT, et récupération du résultat
- Lignes 14-19 : parcours des lignes dans le résultat. Pour chaque ligne : récupération, conversion, et affichage des données de chaque colonne.
- Lignes 21-23 : Les méthodes de JDBC peuvent lever une exception de type SQLException (par exemple, getConnection() dans le cas où la connexion à la base de données échoue). Il faut donc gérer cette erreur. Ici, on se contente d'afficher les détails de la remontée de l'exception dans la pile d'exécution dans la console. Selon le contexte, on pourrait imaginer des traitements supplémentaires, comme afficher ces informations dans une interface graphique, les stocker dans un fichier de log, envoyer une notification, etc.
- Lignes 24-39 : C'est un point essentiel de ce code : **garantir la bonne libération ressources**. Ici, il s'agit à la fois de ressources réseau (connexion à la base de données) et mémoire (stockage des résultats). La libération des ressources est faite via l'appel aux méthodes close() des interfaces JDBC. Le fait de mettre ces appels dans un bloc finally assure qu'ils

seront effectués dans tous les cas (qu'une exception ait été levée ou non). A noter que l'on peut obtenir un code plus concis en utilisant un `try-with-resources`.

Copiez ce code dans votre `main()` et lancez votre programme. Vous devriez obtenir un affichage de la forme :

```
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 électrogène 2
9 Raton laveur 24
10 Aspirateur 58
11 Oscilloscope 12
12 Planche à voile 72
```

#### 4 – Une classe pour interagir avec la base de données

Créez une classe `DatabaseManager` qui possède les caractéristiques suivantes :

- Des attributs `url`, `user` et `password` pour stocker les informations de connexion.
- Un constructeur permettant d'initialiser ces informations.
- Une méthode `printProducts()` qui se connecte à la base de données et liste les produits présents dans la table « articles »

Remplacez le contenu du `main()` par les lignes suivantes :

```
String url = "jdbc:mysql://10.10.57.5/inventaire";
String user = "guest";
String password = "guest";
DatabaseManager dbManager = new DatabaseManager(url, user, password);
dbManager.printProducts();
```

Si votre classe `DatabaseManager` est bien implémentée, l'exécution du `main()` devrait aboutir au même résultat que dans la partie 3.

#### 5 – Une classe pour modéliser un produit

Dans cette partie, nous allons créer une classe `Product` qui modélise un produit indépendamment de la base de données utilisée.

*Intérêt :* Les autres parties de l'application, par exemple l'interface graphique, manipuleront des instances de `Product` sans savoir comment ces produits sont stockés en base. En cas de modification de la table, ou même du type de base de données, ces parties ne seront pas impactées : le seul code à modifier sera celui de `DatabaseManager`.

### Création de la classe

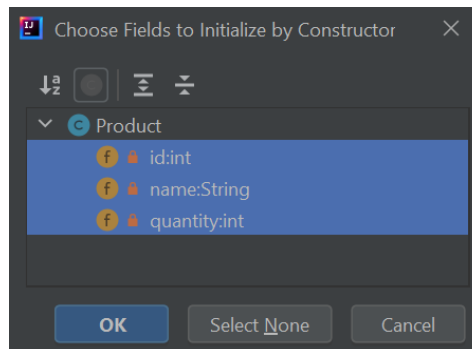
Créez une classe Product ayant les attributs suivants :

Nom	Type	Description
id	int	Identifiant du produit
name	String	Nom du produit
quantity	int	Nombre de produits en stock

Comme cette classe est basique (aucun traitement ni vérification sur les données en entrée ou en sortie), nous allons générer automatiquement le code des constructeurs et des accesseurs.

### Génération du constructeur

- Dans le code, faites un clic droit à l'intérieur de la classe, et cliquez sur « Generate... » (ou appuyez sur ALT+Insert)
- Cliquez sur « Constructor... », une fenêtre affichant les attributs de la classe s'ouvre :



- Sélectionnez tous les attributs en cliquant dessus en maintenant la touche CTRL appuyée.
- Ça y est, votre constructeur est écrit !

### Génération des accesseurs

Faites la même opération, mais en sélectionnant « Getter and Setter » au lieu de « Constructor » dans le menu « Generate ».

Ceci devrait générer 6 accesseurs : 1 en lecture et 1 en écriture pour chaque attribut.

Vous remarquerez que la convention de nommage utilisée est la même que celle présentée en cours : get/set + nom de l'attribut, le tout en camel case.

### Avertissement

La génération automatique de code ne vous dispense pas de savoir l'écrire vous-mêmes : vous n'aurez pas toujours un IDE sous la main ! Dans certains contextes, vous serez peut-être amenés à écrire du Java dans un simple éditeur.

Par ailleurs, à part dans des classes très simples comme celles-ci, le code généré sera souvent à compléter. Néanmoins, il peut constituer une bonne base de travail.

## 6 – Séparation de la récupération et de l’affichage des données

Modifiez votre classe DatabaseManager de la manière suivante :

- Ajoutez une méthode fetchProducts() qui récupère les produits dans la table « article » et remplit la liste products de l’instance. Pour chaque ligne de la table, on devra créer une instance de la classe Product et l’ajouter à la liste products.
- Modifiez la méthode printProducts() affiche les produits de la liste products.

Remplacez le contenu du main() par les lignes suivantes :

```
String url = "jdbc:mysql://10.10.57.5/inventaire";  
String user = "guest";  
String password = "guest";  
DatabaseManager dbManager = new DatabaseManager(url, user, password);  
dbManager.fetchProducts();  
dbManager.printProducts();
```

Exécutez votre programme, vous devriez avoir le même résultat que précédemment. La grande différence est que la récupération des données et leur affichage sont clairement séparées. Ceci permet d’afficher les données plusieurs fois, et potentiellement de plusieurs manières différentes (dans la console, dans une interface graphique) sans solliciter le serveur.

## 7 – Une visualisation style « tableur »

JTable est un composant Swing qui ressemble à ce que l’on peut voir dans un tableur. Il représente des données stockées dans un tableau à 2 dimensions sous la forme d’une table avec laquelle on peut interagir. On peut par exemple trier les données, éditer les cellules, etc.

Pour instancier une JTable, il faut fournir deux informations :

- Un tableau de chaînes de caractères, contenant le nom de chaque colonne.
- Un tableau 2D d’objets (un tableau de tableaux d’objets) contenant les données de la table. La première dimension correspond aux lignes et la deuxième aux colonnes.

Exemple : Le code suivant :

```
Object[][] data = new Object[2][3];  
  
data[0][0] = 1;  
data[0][1] = "Phone";  
data[0][2] = 42;  
  
data[1][0] = 2;  
data[1][1] = "Computer";  
data[1][2] = 5;
```

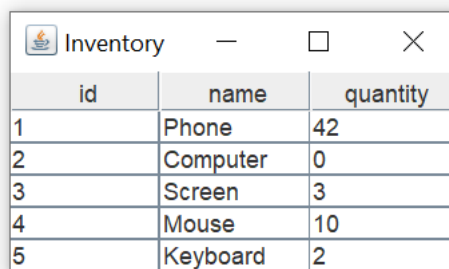
Génère le tableau data suivant :

1	"Phone"	42
2	"Computer"	5

← Une ligne = un produit

Pour vous faire gagner du temps, on vous fournit une classe de démonstration appelée InventoryWindow, qui affiche une JTable dans une JFrame.

Récupérez cette classe dans la page Moodle du cours, intégrez-la à votre projet, et exécutez sa méthode main(). Vous devriez voir apparaître une fenêtre de la forme :



id	name	quantity
1	Phone	42
2	Computer	0
3	Screen	3
4	Mouse	10
5	Keyboard	2

Dans la classe fournie, les données insérées dans la JTable sont des données en dur.

**Votre mission :** modifier ce code pour remplir la JTable avec les données provenant de la base « inventaire ».

*Aide :* Si vous ne voyez pas comment démarrer, voici quelques indications :

- Dans le main(), instancier un DatabaseManager
- Demander au DatabaseManager de mettre à jour la liste des produits en mémoire en consultant la base de données
- Fournir cette liste au constructeur de InventoryWindow
- Dans InventoryWindow, ajouter une méthode qui transforme une liste de produits (LinkedList<Product>) en tableau 2D d'objets (Object[][]) exploitable par la JTable.

*Amélioration facultative :* récupérez dynamiquement les noms de colonne avec JDBC, plutôt que de les mettre en dur. Indication : regarder la méthode [getMetaData\(\)](#) de ResultSet.

## 8 – [Optionnel] Edition des données

Faites en sorte que l'on puisse mettre à jour les quantités de produits en base en éditant la cellule correspondante.

Indications :

- quand une cellule est modifiée, la JTable émet un [TableModelEvent](#), qui peut être écouté en implémentant l'interface [TableModelListener](#).



L'écouteur ne doit pas être ajouté à la JTable directement, mais à son modèle de données.  
Exemple : Si on veut ajouter le TableModelListener `tml` à la JTable `jt`, alors il faut écrire :

```
jt.getModel().add(tml)
```

Dans l'événement, on peut récupérer les coordonnées de la case éditée.

- Pour mettre à jour la base de données, il faut exécuter une requête de type UPDATE, avec la méthode `executeUpdate()` de `ResultSet`, plutôt que `executeQuery()`. Cette méthode renvoie un entier indiquant le nombre de lignes modifiées.
- Une fois la mise à jour effectuée, n'oubliez pas d'actualiser le contenu de la JTable !

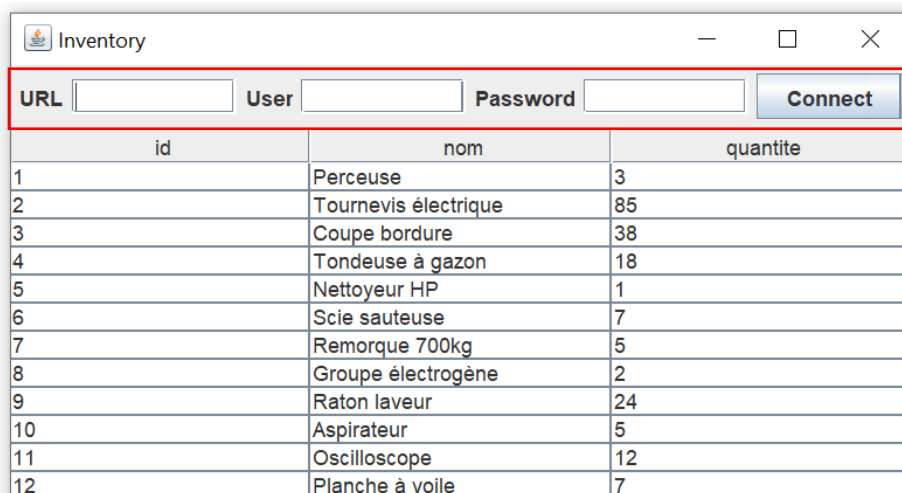
*Améliorations :*

- Si vous avez utilisé une `Statement`, remplacez-la par une `PreparedStatement` : cela protège contre les injections SQL et améliore les performances.
- Faites en sorte que seules les cellules correspondant aux quantités de produits soient éditables.

## 9 – [Optionnel] Saisie des informations de connexion dans l'interface graphique

Ajoutez, en haut de la fenêtre, une partie permettant de se connecter à la base de données, comportant :

- Des champs pour renseigner les informations de connexion
- Un bouton pour établir la connexion avec la base de données



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 électrogène	2
9	Raton laveur	24
10	Aspirateur	5
11	Oscilloscope	12
12	Planche à voile	7

*Amélioration :* Mettez toute cette partie dans un composant graphique `ConnectionPanel`, que vous pourrez inclure dans votre interface en une ligne de code.