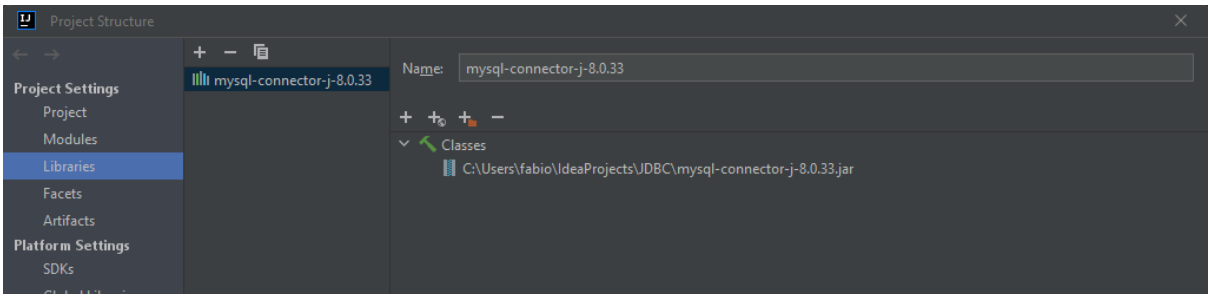
TP guidé JDBC

Le but de ce TP sera d'apprendre à écrire dans une base de données via Java

Partie 1 : Initialiser le projet sur IntelliJ.

Créez un projet Intellij puis ajoutez le JAR dans Project structure>Librairies.



Partie 2 : Créer une base de données Mysql.

En utilisant Xamp/Wamp etc… créer une base de données.

Puis importez le fichier .sql que j'ai mis en piece jointe dans le mail.

Regardez ensuite les tables qui ont été créées

Partie 3 : Créer notre première requête.

Vérifier la connexion à votre base de données grâce à cette requête.

```java
import java.sql.*;
public class Main {
    public static void main(String[] args) {
        try{
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/ecole", user: "root", password: "");

            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery( sql: "select * from tuser");
            while(rs.next()){
                System.out.println(rs.getString( columnIndex: 2));
            }
            con.close();
        }catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

elle doit afficher la liste des élèves dans le terminal

Partie 4 : Créer une requête d'ajout dans la base de données.

```java
import java.sql.*;
public class Main {
    public static void main(String[] args) {
        try{
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/ecole", user: "root", password: "");

            Statement st = con.createStatement();
            var prepareStatement = con.prepareStatement( sql: "INSERT INTO televe(nom,prenom) VALUES('theo','dupont');");
            prepareStatement.executeUpdate();
            con.close();
        }catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Partie 5 : Créer notre modèle d'objet.

Nous allons maintenant créer un objet Eleve ayant pour attributs :

- nom
- prenom
- id

créez un constructeur avec nom et prenom en parametre

puis des getters et setters pour nom et prenom.

Partie 6 : Préparer des requêtes en lien avec notre objet.

Nous allons créer une classe EleveDAO qui consistera à proposer un ensemble de fonctions d'accès à la table eleve.

Cette classe va nous permettre de :

- lister les élèves
- ajouter un élève
- supprimer un élève
- modifier un élève

voici un exemple pour les deux premières fonctions

```java
public List<Eleve> getEleves(){
    List<Eleve> eleves = new ArrayList<>();
    try{
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/ecole", user: "root", password: "");

        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery( sql: "select * from televe");
        while(rs.next()){
            eleves.add(new Eleve(rs.getString( columnIndex: 1),rs.getString( columnIndex: 2)));
        }
        con.close();
    }catch (Exception e) {
        System.out.println(e);
    }
    return eleves;
}
```

```java
public void ajouterEleve(Eleve eleve){
    try{
        Class.forName( className: "com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/ecole", user: "root", password: "");

        PreparedStatement statement = con.prepareStatement( sql: "INSERT INTO televe(nom,prenom) VALUES(?,?);");
        statement.setString( parameterIndex: 1,eleve.getNom());
        statement.setString( parameterIndex: 2,eleve.getPrenom());

        statement.executeUpdate();

        con.close();
    }catch (Exception e) {
        System.out.println(e);
    }
}
```

et voici un exemple d'utilisation dans le main :

```java
public class Main {
    public static void main(String[] args) {
        afficherEleves();
        new EleveDAO().ajouterEleve(new Eleve( nom: "test", prenom: "test"));
        afficherEleves();

    }

    2 usages
    public static void afficherEleves(){
        List<Eleve> eleves= new EleveDAO().getEleves();
        eleves.stream().forEach(eleve -> System.out.println(eleve.toString()));
    }
}
```

petit rappel SQL pour modifier une ligne d'un tableau :

Exemple

Imaginons une table "client" qui présente les coordonnées de clients.

Table "client" :

id	nom	rue	ville	code_postal	pays
1	Chantal	12 Avenue du Petit Trianon	Puteaux	92800	France
2	Pierre	18 Rue de l'Allier	Ponthion	51300	France
3	Romain	3 Chemin du Chiron	Trévérien	35190	France

Modifier une ligne

Pour modifier l'adresse du client Pierre, il est possible d'utiliser la requête SQL suivante :

```
UPDATE client
SET rue = '49 Rue Ameline',
  ville = 'Saint-Eustache-la-Forêt',
  code_postal = '76210'
WHERE id = 2
```

Cette requête sert à définir la colonne rue à "49 Rue Ameline", la ville à "Saint-Eustache-la-Forêt" et le code postal à "76210" uniquement pour ligne où l'identifiant est égal à 2.

Partie 7 : Faire évoluer notre modèle.

Comme vous l'avez surement vu, il existe également un table note.

Comme pour les eleves, créez une classe note avec une enumeration pour les matières

une enumeration se definit comme cela :

```
public enum Matiere{
MATHS,ANGLAIS,PHYSIQUE,INFORMATIQUE;
}
```

Après avoir implémenté la dao nes notes, ajouter à notre classe élève une liste de notes.

Sachant que la table note à besoin d'un id élève, faites en sorte de lier l'ensemble et faire évoluer la dao élève.

L'application finale doit pouvoir :

- **ajouter un élève**
- **lui ajouter une note**
- **afficher la liste des notes d'un élève**
- **afficher la liste des élèves ainsi que leurs notes.**