Exploration

Accéder à une base de données mySQL en Java

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision

Sommaire

Nomenclature :

- **Assimiler** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découvrir** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Voyager** : Projet en autonomie totale. Environnement ouvert : Vous êtes le capitaine !

A Introduction

L'utilisation d'une base de données est possible en Java, cependant elle nécessite l'usage d'un connecteur. Il s'agit d'une bibliothèque qui va adapter les chaînes et objets Java au format MySQL.

Ce connecteur est appelé JDBC (Java DataBase Connector). Il s'agit d'une API Java qui permet d'utiliser le même code, quel que soit le moteur de base de données (MySQL, MariaDB, PostGreSQL, Oracle…)

Intérêt de l'API JDBC

L'API a plusieurs rôles :

- **Indépendance de la plateforme** : JDBC est une API standard de Java qui permet aux applications Java de se connecter à des bases de données sans dépendre de la plateforme spécifique de la base de données.
- **Facilité d'utilisation** : JDBC fournit une interface standardisée pour exécuter des requêtes SQL, gérer les transactions, et récupérer les résultats des requêtes.
- **Performance** : JDBC offre des fonctionnalités avancées pour optimiser les performances, comme le pool de connexions, qui permet de réutiliser les connexions existantes plutôt que de créer une nouvelle connexion pour chaque requête.
- **Sécurité** : JDBC prend en charge les fonctionnalités de sécurité telles que l'authentification, le chiffrement des données et la gestion des permissions.
- **Portabilité** : Les applications Java utilisant JDBC peuvent être déployées sur n'importe quelle plateforme qui supporte Java, ce qui rend le développement et le déploiement plus flexibles et portables.

A.1 Prérequis

Il faut télécharger le connecteur sur le site mySQL : https://dev.mysql.com/downloads/connector/j/

Sélectionnez "Platform independent" et cliquez sur le lien au format ZIP.

Le fichier sera à décompresser à l'intérieur de notre projet (dans un premier temps).

B Programme de lecture de la base

Nous allons utiliser la base de données des IMC pour cette activité. Si vous ne l'avez pas déjà créé, le code SQL est disponible en annexe.

B.1 Prérequis

Aucun

C Création du projet

Le projet sera réalisé en Java avec une interface graphique.

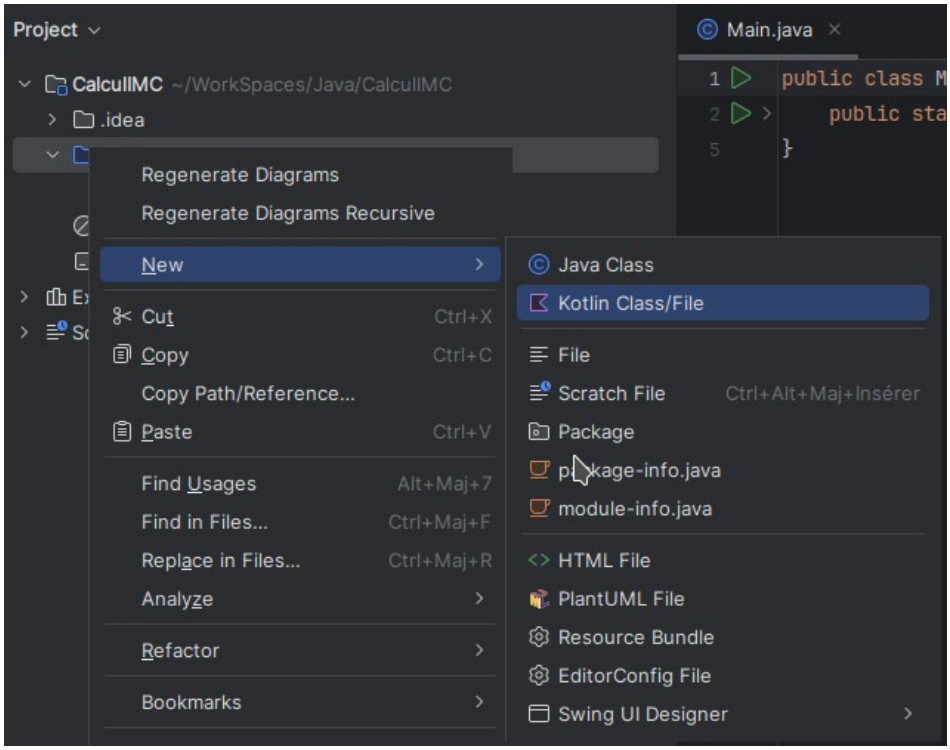
C.1 Création du projet

Créez un nouveau projet de type Java, en IntelliJ :

File → New → Project

Nom : **CalculIMC**
Build System : **IntelliJ**

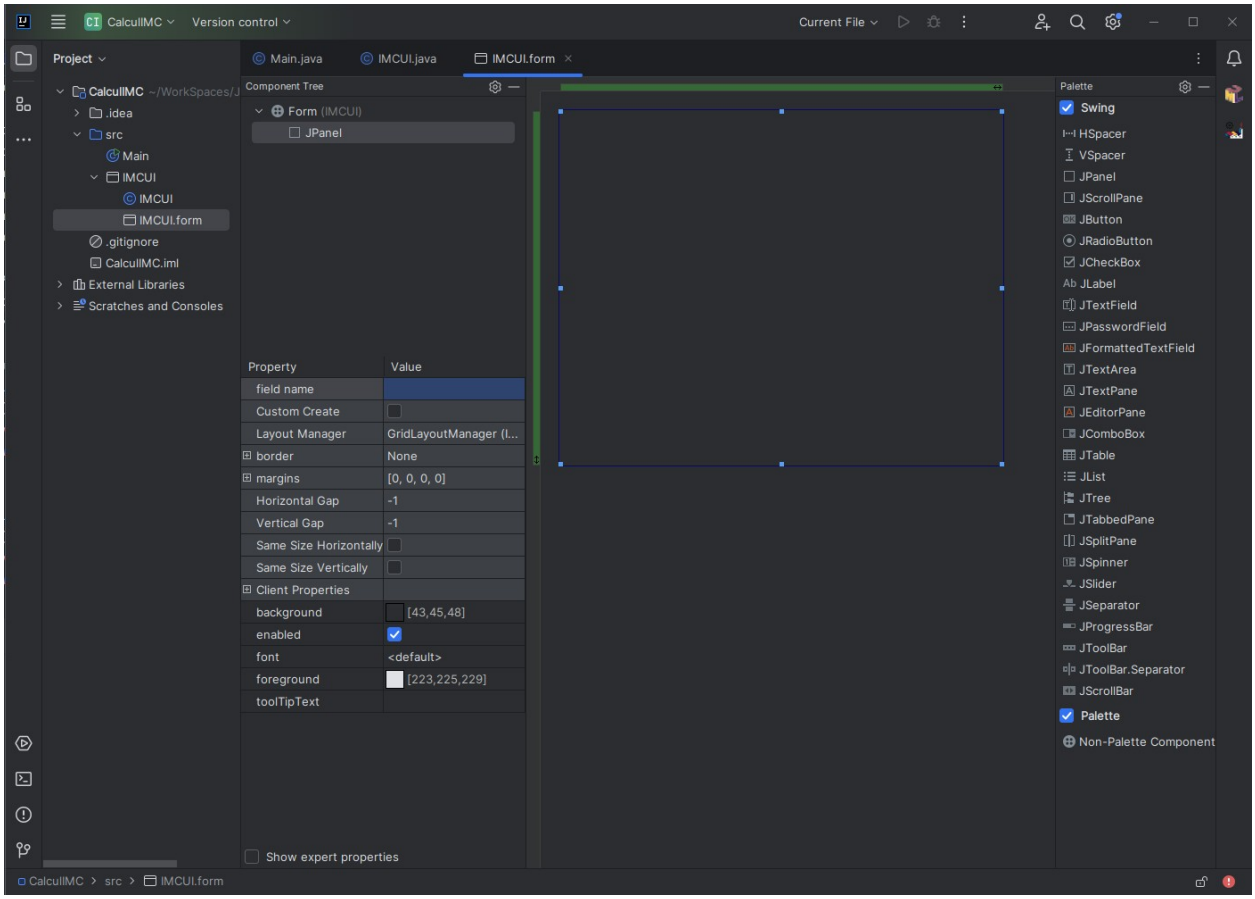
Ensuite, dans le répertoire **source**, faire un clic-droit, **New → Swing UI Designer → GUI Form**



Et créer une forme appelée **IMCGUI**. Il existe différents *layouts* pour les interfaces, nous allons retenir **GridBagLayout**.

C.2 Création interface graphique

L'interface d'IntelliJ change pour permettre une édition graphique des objets.



Le carré représente la fenêtre, nous allons y ajouter nos champs :

`JLabel` permet d'écrire du texte dans la fenêtre,

`JTextField` est un champ de saisie,

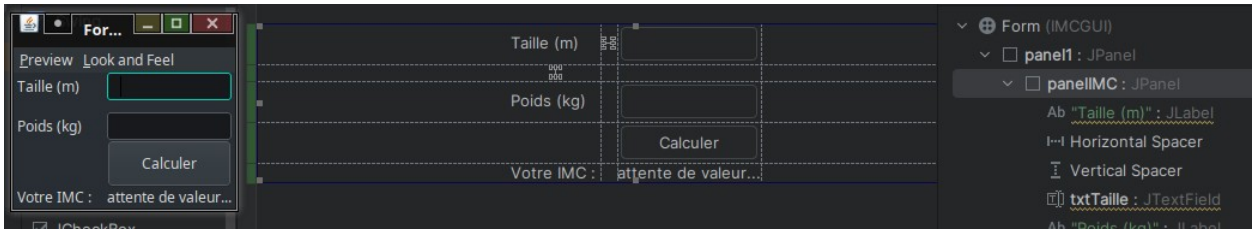
`JButton` est un bouton d'action.

Il est important de donner des noms aux objets dont nous allons nous servir avec le code : **txtTaille** et **txtPoids** seront utilisés pour les `JTextField` de saisie du poids et de la taille.

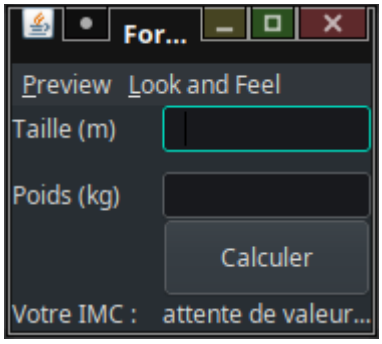
Le bouton [Calculer] aura le nom **btnCalculer**.

Enfin, le résultat sera un `JLabel` portant le nom de **txtResultat**.

Voici un exemple de résultat : vous pouvez tester à tout moment ce qui se passe en faisant un clic droit dans une zone libre de l'éditeur de forme et choisissant **Preview**.



⚠ Votre affichage peut différer sensiblement du mien, car j'utilise Linux avec une interface KDE : il est probable que sous Windows, l'affichage soit plus classique.



Attention : Il faut ajouter `extends JFrame` dans le fichier IMCGUI.java

```
public class IMCGUI extends JFrame {
```

C.3 Utilisation de la classe depuis Main()

Comme vous savez utiliser les classes, vous comprenez qu'il est plus intéressant de séparer le programme principal (main()) des classes d'affichage et de les appeler en les instanciant.

C'est ce que nous allons faire, dans le fichier Main.java :

- Importer le système graphique de Java (swing)
- Appeler la classe IMCGUI
- Indiquer le panneau à afficher
- Donner un titre et une dimension à notre forme
- La rendre visible
- Autoriser la fermeture du programme lors de la fermeture de la forme

Voici les modifications :

main.java

```
import javax.swing.*;

public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
        IMCGUI form = new IMCGUI();
        form.setContentPane(form.panelIMC);
        form.setTitle("Calcul IMC");
        form.setSize(250, 200);

        form.setVisible(true);
        form.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Désormais, il est possible de compiler le programme et normalement, la fenêtre doit s'afficher.

C.4 Ajouter des interactions

Il faut maintenant interagir avec la forme, et notamment le bouton [Calculer]. Pour cela, dans IMCGUI.form, faire un clic droit sur le bouton et choisir "Create listener" → "ActionListener"

Le constructeur de la classe IMCGUI contient maintenant le code suivant :

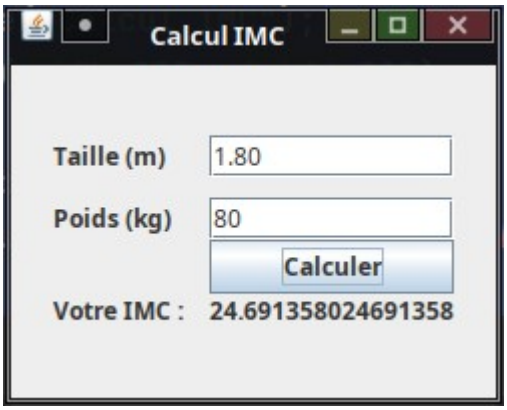
```
    public IMCGUI() {
        btnCalculer.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
            }
        });
    }
```

Il suffit d'ajouter les actions à faire, cette partie vous revient :

```
    public IMCGUI() {
        btnCalculer.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                Double taille = Double.parseDouble(txtTaille.getText());
                Double poids = Double.parseDouble(txtPoids.getText());
                Double imc = poids/(taille*taille);
                txtResultat.setText(String.valueOf(imc));
            }
        });
    }
```

Il faut noter que les champs txtTaille et txtPoids contenant du texte, il faut le convertir avant d'effectuer un calcul, comme en Javascript (ce dernier, était capable d'effectuer certaines conversions, seul).

Le résultat est le suivant :



Afin d'être plus propre, nous allons simplement arrondir le résultat :

```
Double imc = poids/(taille*taille);
imc = Math.round(imc * 100.0) / 100.0;
```

Gestion d'erreur (exception)

Enfin, pour rendre le programme plus fiable, nous introduisons la gestion des erreurs, ici de manière simplifiée.

```
    public IMCGUI() {
        btnCalculer.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    Double taille = Double.parseDouble(txtTaille.getText());
                    Double poids = Double.parseDouble(txtPoids.getText());
                    Double imc = poids/(taille*taille);
                    imc = Math.round(imc * 100.0) / 100.0;
                    txtResultat.setText(String.valueOf(imc));
                } catch (NumberFormatException ex) {
                    txtResultat.setText("Erreur de saisie");
                }
            }
        });
    }
```

Le programme est désormais robuste et opérationnel.

D Gestion de la base de données

D.1 Ajout du champ Nom dans le formulaire

Il faut éditer la forme pour ajouter un `txtField` que vous nommerez `txtNom`.

Puis, dans l'action listener, il faut récupérer ce champ, au-dessus de la récupération de la taille et du poids :

```
String nom = txtNom.getText();
```

Enfin, il faut appeler une méthode (qui sera locale pour le moment) pour écrire les valeurs dans la base de données.

```
// Enregistrer les données dans la base de données
enregistrerIMC(nom, poids, taille);
```

D.2 Méthode d'enregistrement IMC

L'idée est ici de communiquer vers la base distante et y insérer les données récoltées.

Toujours dans IMCGUI.java, insérez le code suivant après la méthode IMVGUI() :

IMCGUI.java

```
    private void enregistrerIMC(String nom, Double poids, Double taille) {
        String url = "jdbc:mysql://localhost:3306/DBIMC";
        String user = "votre_utilisateur";
        String password = "votre_mot_de_passe";

        try (Connection conn = DriverManager.getConnection(url, user, password);
             PreparedStatement stmt = conn.prepareStatement("INSERT INTO imc (nom, poids, taille)
VALUES (?, ?, ?)")) {
            stmt.setString(1, nom);
            stmt.setDouble(2, poids);
            stmt.setDouble(3, taille);
            stmt.executeUpdate();
            System.out.println("Données enregistrées dans la base de données.");
        } catch (SQLException ex) {
            System.out.println("Erreur lors de l'enregistrement dans la base de données : " +
ex.getMessage());
        }
    }
```

Pour le moment, de nombreuses parties sont en erreur (Connexion

Insérez le code suivant en entête :

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
```

E Annexes

E.1 Sources

E.1.1 Base pour IMC

Création d'une base IMC contenant un id, le nom, le poids et la taille d'une personne.

```
CREATE TABLE IMC (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    poids DECIMAL(5,2), -- Le poids en kg
    taille DECIMAL(5,2) -- La taille en mètres
);
```

E.1.2 Remplissage de quelques valeurs

Quelques personnages pour tester la base :

```
INSERT INTO IMC (nom, poids, taille) VALUES
('NORRIS Chuck', 80.5, 1.75),
('Spock', 70.2, 1.85),
('Ironman', 85.0, 1.80),
('Batman', 95.5, 1.90);
```

E.2 Autres

Les différents layout en Java Swing

https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html