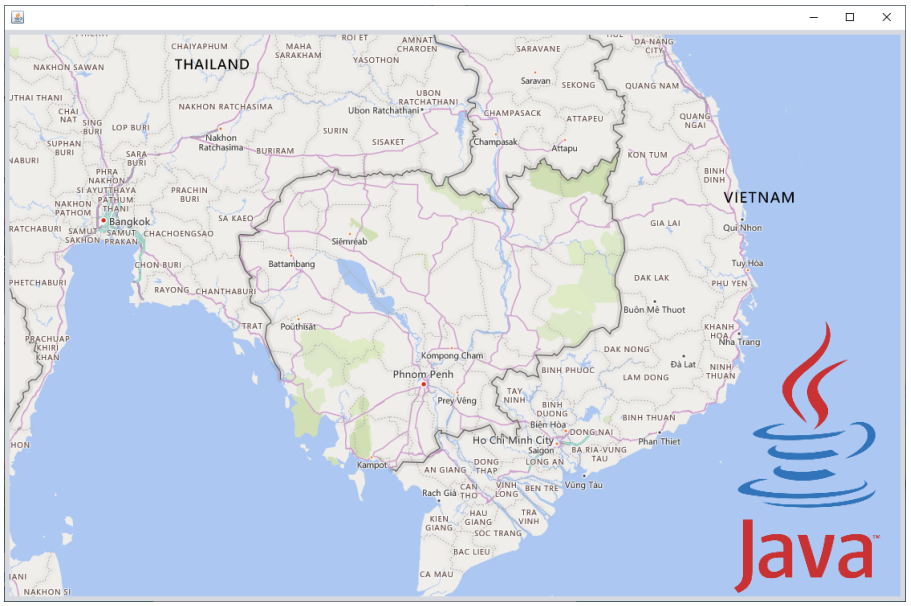


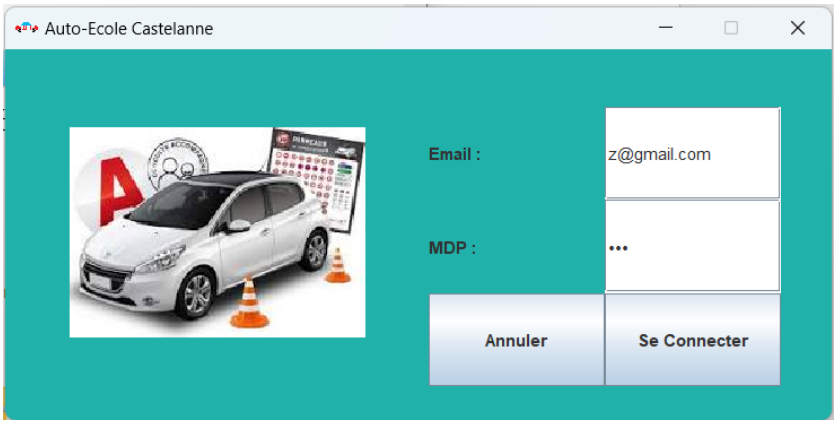
JMapViewer

Table des matières

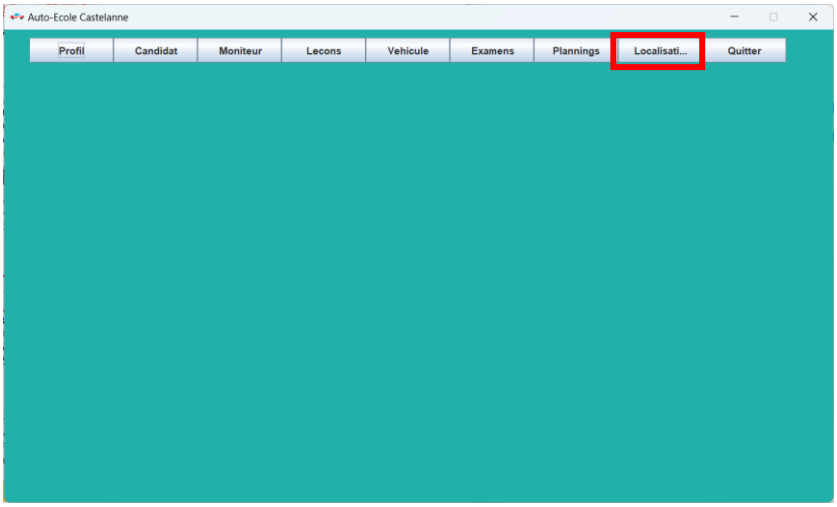
Introduction

Ce programme Java affiche une interface graphique avec une carte interactive et une liste d'utilisateurs récupérés à partir d'une base de données MySQL. Chaque utilisateur est associé à une adresse, et lorsque l'utilisateur est sélectionné, sa position est affichée sur une carte OpenStreetMap à l'aide du composant `JMapViewer`.

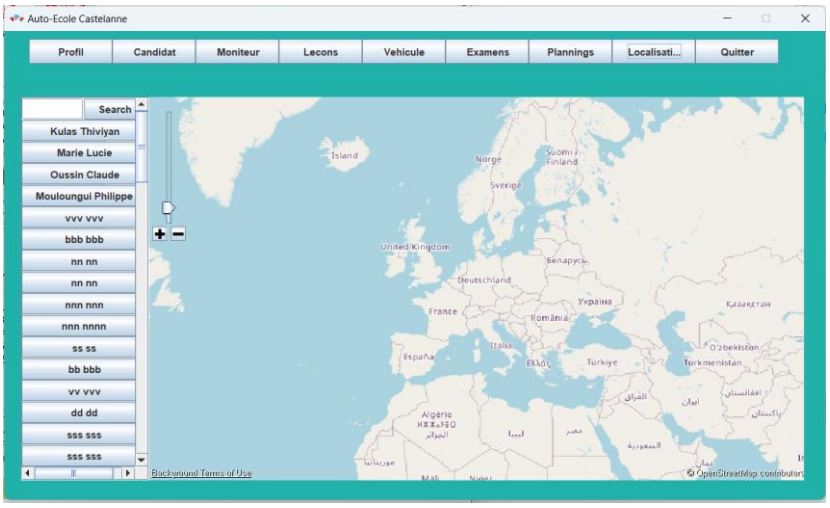
Je me connecte sur le client lourd.



Puis je clique sur « localisation »



On voit la carte de localisation.



Pré-requis

Avant d'exécuter ce programme, assurez-vous d'avoir les prérequis suivants :

1. Java Development Kit (JDK) 8 ou supérieur.
2. Base de données MySQL installée avec les tables requises.
3. Les bibliothèques suivantes doivent être incluses dans votre projet :
    - JMapViewer
    - JSON (pour la manipulation des données JSON)
    - JDBC Driver pour MySQL (pour la connexion à la base de données)

1. Comment installer JMapView

Pour créer une carte avec localisation en utilisant Java Swing dans Eclipse, on peut utiliser plusieurs bibliothèques Java. L'une des plus populaires est JMapViewer, qui est simple à utiliser et fournit des fonctionnalités de base pour afficher des cartes et des marqueurs.

Pour avoir JMapViewer, il faut téléchargez JMapViewer depuis le site officiel ou utilisez un gestionnaire de dépendances comme Maven ou Gradle.

Voici le lien pour télécharger JMapViewer :
https://mvnrepository.com/artifact/org.openstreetmap.jmapviewer/jmapviewer/2.16

1. 2. Comment installer JSON

Pour manipuler le donnée des utilisateurs qui est dans la base de donnée, il faut installer « org.json »

Voici le lien pour télécharger org.json :
https://mvnrepository.com/artifact/org.json/json/20240303

Structure du code

Le code est organisé principalement en une seule classe :

- `MapExample` : Cette classe gère l'interface graphique et les interactions avec la base de données ainsi qu'avec l'API de géocodage.

```
public class MapExample extends JFrame {}
```

Fonctionnalités principales

Ce programme fournit les fonctionnalités suivantes :

1. Affichage des utilisateurs depuis une base de données MySQL : Les utilisateurs sont affichés sous forme de boutons dynamiques sur une interface graphique.
2. Géocodage d'adresses : Conversion des adresses des utilisateurs en coordonnées géographiques (latitude et longitude) à l'aide de l'API Nominatim d'OpenStreetMap.
3. Affichage de la carte interactive : Utilisation de `JMapViewer` pour afficher une carte avec des marqueurs indiquant les emplacements des utilisateurs.
4. Interaction utilisateur : Lorsqu'un bouton représentant un utilisateur est cliqué, la carte se centre sur l'adresse de cet utilisateur et ajoute un marqueur à cet endroit.

Fonctionnement du code

1. Initialisation de la carte

La carte est créée à l'aide de la classe `JMapViewer`. Elle est initialisée avec une source de tuiles OpenStreetMap et ajoutée à l'interface utilisateur principale.

```
private JMapViewer map;
```

2. Chargement des utilisateurs

```
64    private void loadUsers(JPanel panel) {
65        Connection connection = null;
66        Statement statement = null;
67        ResultSet resultSet = null;
68
69        try {
70            // Établir la connexion
71            connection = DriverManager.getConnection(jdbcUrl, username, password);
72
73            // Créer une déclaration
74            statement = connection.createStatement();
75
76            // Exécuter la requête SQL pour récupérer les utilisateurs
77            String sql = "SELECT IDUSER, NOM, PRENOM, ADRESSE FROM USER";
78            resultSet = statement.executeQuery(sql);
79
80            // Parcourir les résultats et créer des boutons pour chaque utilisateur
81            while (resultSet.next()) {
82                int idUser = resultSet.getInt("IDUSER");
83                String nom = resultSet.getString("NOM");
84                String prenom = resultSet.getString("PRENOM");
85                String adresse = resultSet.getString("ADRESSE");
86
87                // Ajouter l'adresse au map
88                userAddresses.put(nom + " " + prenom, adresse);
89
90                // Créer un bouton pour chaque utilisateur
91                JButton userButton = new JButton(nom + " " + prenom);
92                userButton.addActionListener(new ActionListener() {
93                    @Override
94                    public void actionPerformed(ActionEvent e) {
95                        showUserLocation(adresse);
96                    }
97                });
98                panel.add(userButton);
99            }
100
101       } catch (SQLException e) {
102           e.printStackTrace();
103       } finally {
104           try {
105               // Fermer les ressources
106               if (resultSet != null) resultSet.close();
107               if (statement != null) statement.close();
108               if (connection != null) connection.close();
109           } catch (SQLException e) {
110               e.printStackTrace();
111           }
112       }
113   }
```

Pour charger l'utilisateur, il faut la méthode loadUsers sont le rôle est :

- Se connecte à la base de données MySQL.
- Exécute une requête SQL pour récupérer les informations des utilisateurs (ID, nom, prénom, adresse).
- Crée un bouton pour chaque utilisateur, et ajoute ces boutons dynamiquement à l'interface pour l'adresse des utilisateurs.

4. Recherche de localisation

```java
private void showUserLocation(String address) {
    double[] coordinates = getCoordinates(address);
    if (coordinates[0] != 0 && coordinates[1] != 0) {
        map.setDisplayPosition(new Coordinate(coordinates[0], coordinates[1]), 15);
        map.getMapMarkerList().clear(); // Clear existing markers
        MapMarker marker = new MapMarkerDot(address, new Coordinate(coordinates[0], coordinates[1]));
        map.addMapMarker(marker);
    } else {
        System.err.println("Adresse introuvable: " + address);
    }
}
```

Pour localiser l'utilisateur, il faut la méthode `showUserLocation` qui doit déclencher lorsqu'un utilisateur clique sur un bouton. Son rôle est :

- Appelle la méthode `getCoordinates` pour convertir l'adresse en latitude et longitude.
- Affiche ces coordonnées sur la carte et recentre la vue sur la position spécifiée.

5. Géocodage des adresses

```java
public static double[] getCoordinates(String address) {
    double[] coordinates = new double[2];
    try {
        String url = String.format("https://nominatim.openstreetmap.org/search?q=%s&format=json&addressdetails=1", address.replace(" ", "+"));
        HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("User-Agent", "Mozilla/5.0");
        connection.setRequestProperty("Accept", "application/json");

        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuilder response = new StringBuilder();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();

        JSONArray jsonArray = new JSONArray(response.toString());
        if (jsonArray.length() > 0) {
            JSONObject jsonObject = jsonArray.getJSONObject(0);
            coordinates[0] = jsonObject.getDouble("lat");
            coordinates[1] = jsonObject.getDouble("lon");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return coordinates;
}
```

La méthode `getCoordinates` :

- Envoie une requête HTTP à l'API Nominatim d'OpenStreetMap pour convertir l'adresse en coordonnées géographiques.
- Parse la réponse JSON pour extraire la latitude et la longitude.

Dépendances

Le programme dépend des bibliothèques suivantes :

- Java Standard Libraries : `java.sql`, `javax.swing`, `java.awt`, etc.
- JMapViewer : pour l'affichage de la carte.
- JSON : pour le traitement des données JSON.

- JDBC Driver pour MySQL : pour interagir avec la base de données.

Détails techniques

Configuration de la base de données

La connexion à la base de données est définie avec les paramètres suivants :

- URL : `jdbc:mysql://localhost:3306/auto_ecole_24`
- Nom d'utilisateur : `root`
- Mot de passe : `""` (pas de mot de passe)

La requête SQL suivante est utilisée pour récupérer les données des utilisateurs :

« SELECT IDUSER, NOM, PRENOM, ADRESSE FROM USER; »

Requête API pour le géocodage

Le programme utilise l'API Nominatim d'OpenStreetMap pour convertir une adresse en coordonnées géographiques. La requête est formulée comme suit : https://nominatim.openstreetmap.org/search?q=adresse&format=json&addressdetails

Gestion des erreurs

Le code gère les exceptions SQL et les erreurs de connexion réseau pour l'API Nominatim, avec des messages d'erreur affichés dans la console.

Améliorations possibles

1. Sécurité : Le mot de passe de la base de données devrait être sécurisé (non stocké en clair dans le code).
2. Gestion des erreurs : Améliorer la gestion des erreurs pour donner des messages d'erreur plus clairs et informatifs.
3. Interface utilisateur : Améliorer la disposition et le style de l'interface utilisateur pour une meilleure expérience.
4. Optimisation des requêtes : Ajouter une mise en cache pour éviter les appels redondants à l'API de géocodage.

5. Compatibilité : Permettre l'intégration avec d'autres sources de données pour plus de flexibilité.

Ce document couvre tous les aspects techniques et fonctionnels du programme. Pour toute question ou amélioration supplémentaire, veuillez ajuster le code en conséquence.