Ecole Supérieure
d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

Chapitre 8: POO avancée - JDBC

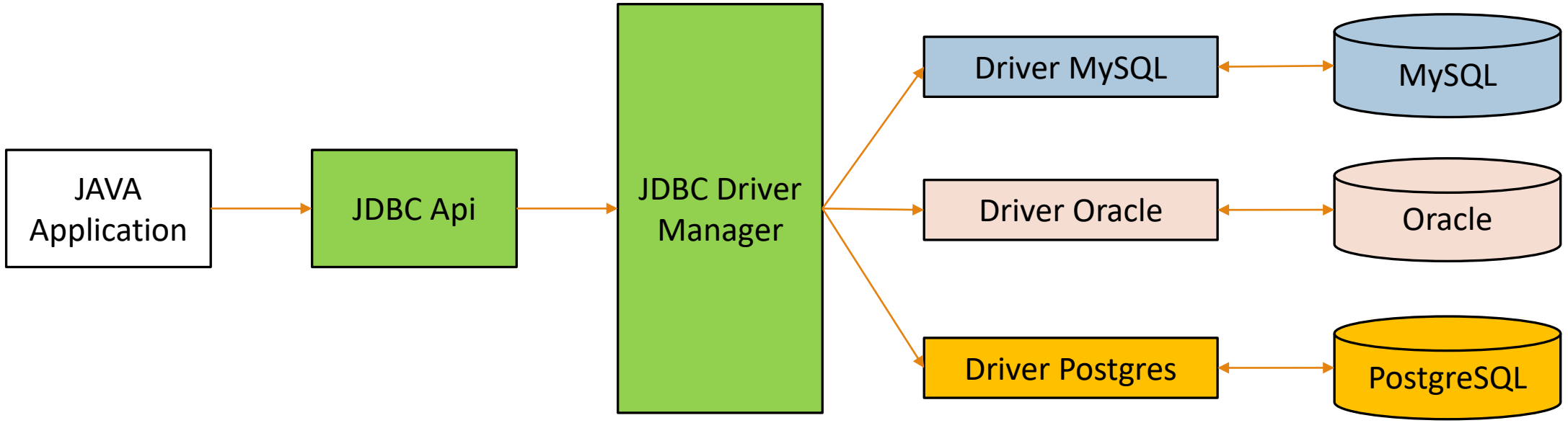
MEHDI NAJIB

Plan

I. Etablir une connexion JDBC
II. Singleton de la connexion
III. Exemple d’insertion
IV. Exemple de Listing
V. Exemple de refresh des JTable

JDBC

JAVA DataBase Connectivity est une interface qui permet d'établir une connexion avec les bases de données en JAVA,



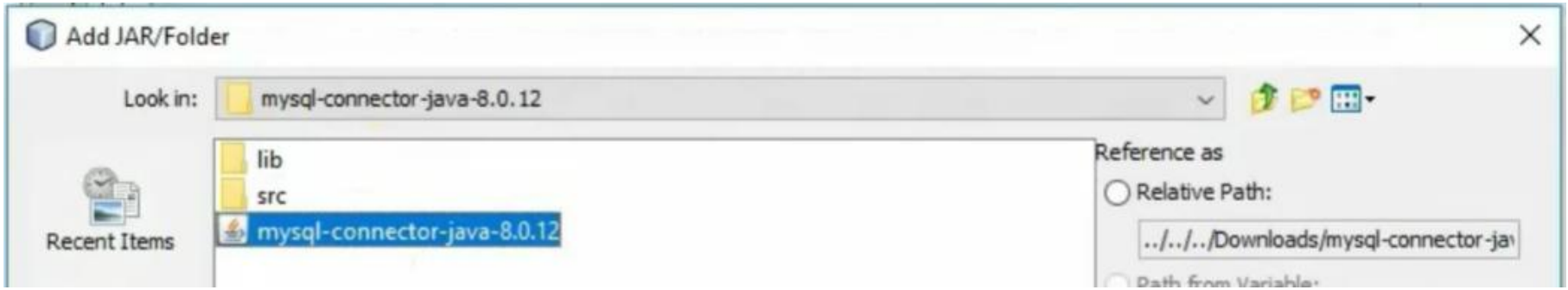
Le driver MySQL

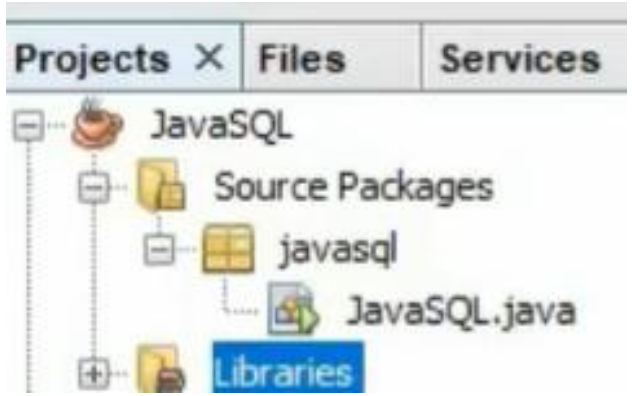
Pour établir une connexion JDBC, il faut utiliser ajouter le driver dans le projet

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

Le driver MySQL

Exemple projet JAVA de base : Ajouter le Jar qui représente le driver dans les librairies du projet



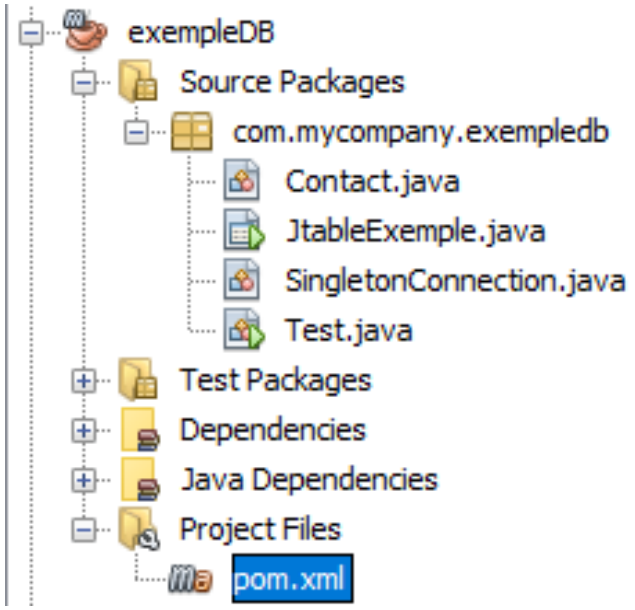


Le driver MySQL (Maven)

Utilisation des dependecies pour une préparation automatique du driver

Modification dans le fichier Pom.XML

```
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.36</version>
  </dependency>
</dependencies>
```



La connexion JDBC

Pour établir une connexion avec la base de données:

1. Serveur DB: 127.0.0.1 (localhost)
2. Port : 3306
3. Nom de la BD: bdJAVA
4. Login: root
5. Password : vide (par défaut)

```
conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/bdJAVA?" +
    "user=root&password=");
```

La connexion JDBC

Exemple complet de la méthode de connexion, avec gestion des erreurs

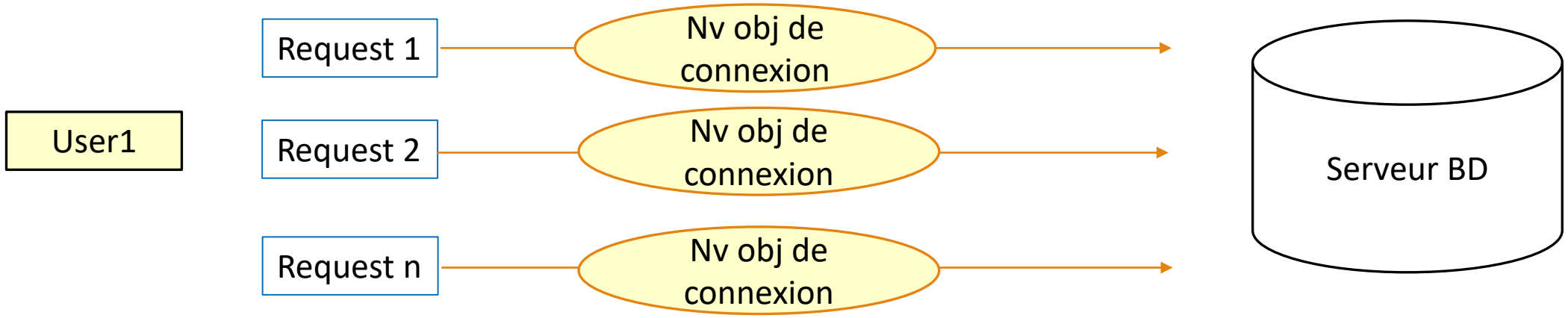
```
public static java.sql.Connection connect() throws SQLException{
        java.sql.Connection conn = null;
        try{
            conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/bdJAVA?" +
                    "user=root&password=");
        }
        catch (SQLException e) {
            System.out.println("SQLException: " + e.getMessage());
            System.out.println("SQLState: "     + e.getSQLState());
            System.out.println("VendorError: "  + e.getErrorCode());
        }
        return conn;
}
```

Singleton de connexion

Un singleton permet d'avoir une et une seule instance d'une classe en JAVA

Un singleton permet d'éviter l'instanciation de plusieurs objets de connexion

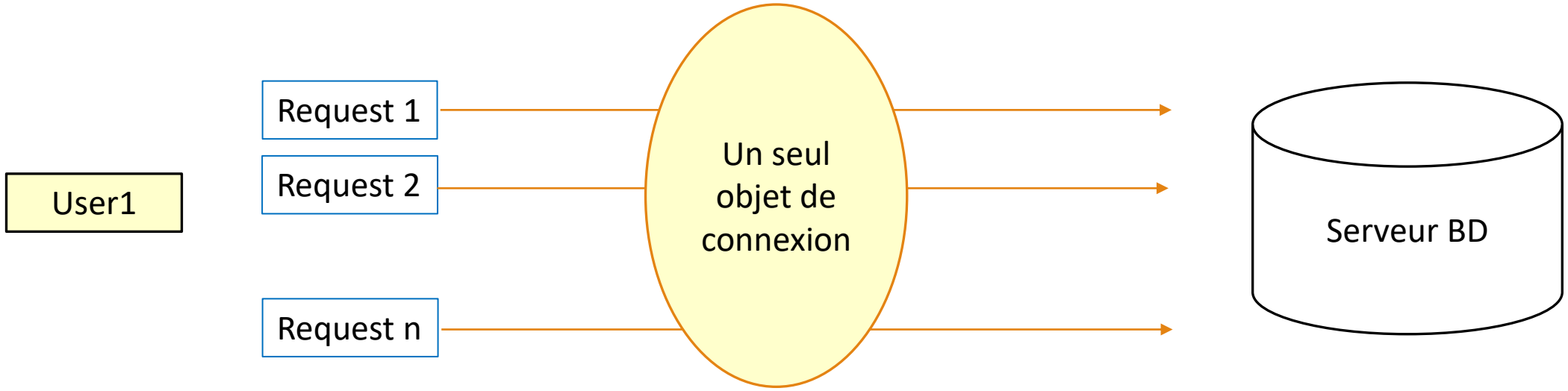
Un singleton permet un bon fonctionnement du serveur de BD,



Singleton de connexion

Un seul objet de connexion :

1. La première exécution on crée l'objet de connexion à la base de données
2. Les autres requêtes réutilisent l'ancien objet de connexion

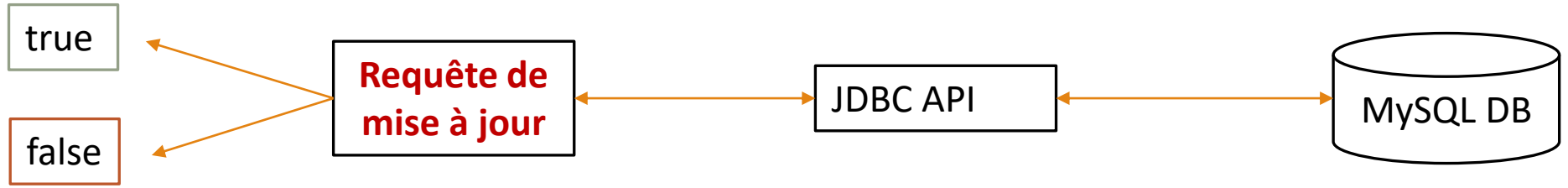


Exemple de Singleton de connexion

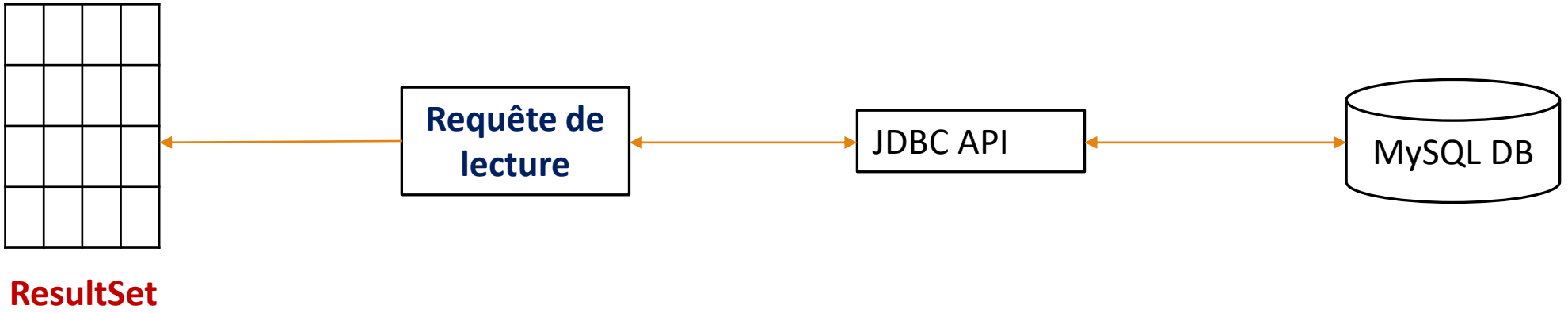
```
import java.sql.DriverManager;
import java.sql.Connection;

public class SingletonConnection {
	private static Connection connection;

	static{
		try {
			Class.forName("com.mysql.jdbc.Driver");
			connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/coursbd", "root", "");
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
	public static Connection getConnection(){
		return connection;
	}
}
```

Deux types de requêtes

- **Requête de mise à jour :**



- **Requête de lecture :**



Insertion dans une Table

La table dans la base de données MYSQL

Insertion dans une Table

Exemple de Bean/Entity

```java
public class Contact {
    int id;
    String pseudo, phone;

    public Contact(String pseudo, String phone) {
        this.pseudo = pseudo;
        this.phone = phone;
    }

    public Contact(int id, String pseudo, String phone) {
        this.id = id;
        this.pseudo = pseudo;
        this.phone = phone;
    }
}
```

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐	1	id	int(11)			Non	*Aucun(e)*		AUTO_INCREMENT	Modifier Supprimer Plus
☐	2	pseudo	varchar(20)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus
☐	3	phone	varchar(20)	utf8mb4_general_ci		Non	*Aucun(e)*			Modifier Supprimer Plus

Insertion dans une Table

Les étapes d'exécution d'une requête de mise à jour

- **Connexion**

```
Connection conn = SingletonConnection.getConnection();
```

- **Prepared statement**

```
String SQL = "INSERT INTO contacts(id, pseudo, phone)  VALUES(null,?, ?)";
PreparedStatement pstmt = (PreparedStatement) conn.prepareStatement(SQL);
```

- **Setting attributes**

```
pstmt.setString(1, c.pseudo);
pstmt.setString(2, c.phone);
```

- **ExecuteUpdate**

```
int r = pstmt.executeUpdate();
```

In
ur

POU
REC

```java
public static boolean enregisterContact(Contact c ){
    boolean res = false;
    int r = 0;
    try {
        Connection conn = SingletonConnection.getConnection();
        String SQL = "INSERT INTO contacts(id, pseudo, phone) "
                   + "VALUES(null,?, ?)";
        PreparedStatement pstmt = (PreparedStatement) conn.prepareStatement(SQL);

        pstmt.setString(1, c.pseudo);
        pstmt.setString(2, c.phone);
        r = pstmt.executeUpdate();
        if (r == 1)
            res= true;
    } catch (SQLException ex) {
        System.out.print("Probleme d'ajout du nouveau contcat");
        System.out.println(ex.getMessage());
    }
    return res;
```

Insertion dans une Table

Liaison avec un événement action performed

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    String pseudo = jtfPseudo.getText();
    String phone = jtfPhone.getText();

    Contact c = new Contact(pseudo, phone);

    boolean resAjt = Contact.enregisterContact(c);

    if(resAjt){
        JOptionPane d = new JOptionPane();
        d.showMessageDialog( this, "Contact enregistre avec success", "Add contact ok", JOptionPane.PLAIN_MESSAGE) ;
    }else{
        JOptionPane d = new JOptionPane();
        d.showMessageDialog( this, "Problème d'enregistremet du nouveau contact", "Add contact problem", JOptionPane.ERROR_MESSAGE);
    }
}
```

Exemple de requête de lecture

Les étapes pour exécuter une requête de selection

- **Connexion**

```
java.sql.Connection conn = SingletonConnection.getConnection();
```

- **Création du statement**

```
String   req = "SELECT * FROM contacts ";
Statement    stmt  = conn.createStatement();
```

- **ExecuteQuery**

```
ResultSet rs = stmt.executeQuery(req);
```

- **ResultSet fetching**

```
while (rs.next()) {
        // getting attributes
}
```

Exemple de requête de lecture

```
public static LinkedList<Contact> getAllContact(){
        LinkedList<Contact> listeContact= new LinkedList<Contact>();
        try{
                java.sql.Connection conn = SingletonConnection.getConnection();
                String req = "SELECT * FROM contacts ";
                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(req);

                Contact c = null;
                while (rs.next()) {
                        int id = rs.getInt(1);
                        String pseudo = rs.getString(2);
                        String phone = rs.getString(3);

                        c = new Contact(id, pseudo, phone);
                        listeContact.add(c);
                }
                rs.close();
                stmt.close();
                }catch(SQLException e){
                        System.out.println("Problème durant la récupération de la liste "
                                        + "des contacts");
                }
        return listeContact;
}
```

Exemple de requête de lecture avec Jtable

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    LinkedList<Contact> listeContact = Contact.getAllContact();

    Vector<String> VCols = new Vector<String>();
    VCols.add("ID");
    VCols.add("Pseudo ");
    VCols.add("Phone Number");

    // Creation des lignes temp
    Vector<Object> Vligne = new Vector<Object>();
    Vector<Vector<Object>> VData = new Vector<Vector<Object>>();

    for(Contact c:listeContact){
        // iteration sur les objests
        Vligne = new Vector<Object>();
        Vligne.add(c.id);
        Vligne.add(c.pseudo);
        Vligne.add(c.phone);
        // store in VData
        VData.add(Vligne);
    }
    DefaultTableModel modelDyn = new DefaultTableModel(VData, VCols);
    jTable1.setModel(modelDyn);
}
```

Exemple de requête de lecture avec JTable

Exemple de filtrage dynamique

Pour activer une recherche par mots clés, il faut adapter la requête suivante

```
String req = "SELECT * FROM contacts WHERE PSEUDO like '%"+keyword+"%' ";
```

Utiliser l'évenement **KeywordKeyTyped** associé à la JtextField proposée à l'utilisateur pour saisir le mot clé

Exemple de filtrage dynamique

Exemple de la fonction de filtrage (attention problème à tester)

```
private void jtfKeywordKeyTyped(java.awt.event.KeyEvent evt) {
      String keyword = jtfKeyword.getText();
     // filtrage BD
      LinkedList<Contact> listeContact = Contact.filtrage(keyword);
    // rafraichir la jtable
     refreshJtable(listeContact);
  }
```

Exemple de filtrage dynamique

La solution KeyReleased event :

```
private void jtfKeywordKeyReleased (java.awt.event.KeyEvent evt) {
      String keyword = jtfKeyword.getText();
     // filtrage BD
      LinkedList<Contact> listeContact = Contact.filtrage(keyword);
    // rafraichir la jtable
     refreshJtable(listeContact);
  }
```

Exemple de filtrage dynamique

Action de filtrage traitement BD

```
public static LinkedList<Contact> filtrage(String keyword){

        LinkedList<Contact> listeContact= new LinkedList<Contact>();
        try{
                java.sql.Connection conn = SingletonConnection.getConnection();
                String req = "SELECT * FROM contacts WHERE PSEUDO like '%"+keyword+"%' ";
                PreparedStatement pstmt = (PreparedStatement) conn.prepareStatement(req);
                ResultSet rs = pstmt.executeQuery(req);
                Contact c = null;
                while (rs.next()) {
                        int id = rs.getInt(1);
                        String pseudo = rs.getString(2);
                        String phone = rs.getString(3);

                        c = new Contact(id, pseudo, phone);
                        listeContact.add(c);
                }
                rs.close();
                pstmt.close();
                }catch(SQLException e){
                        System.out.println("Problème durant la récupération de la liste "
                                        + "des contacts");
                        e.printStackTrace();
                }
        return listeContact;
}
```

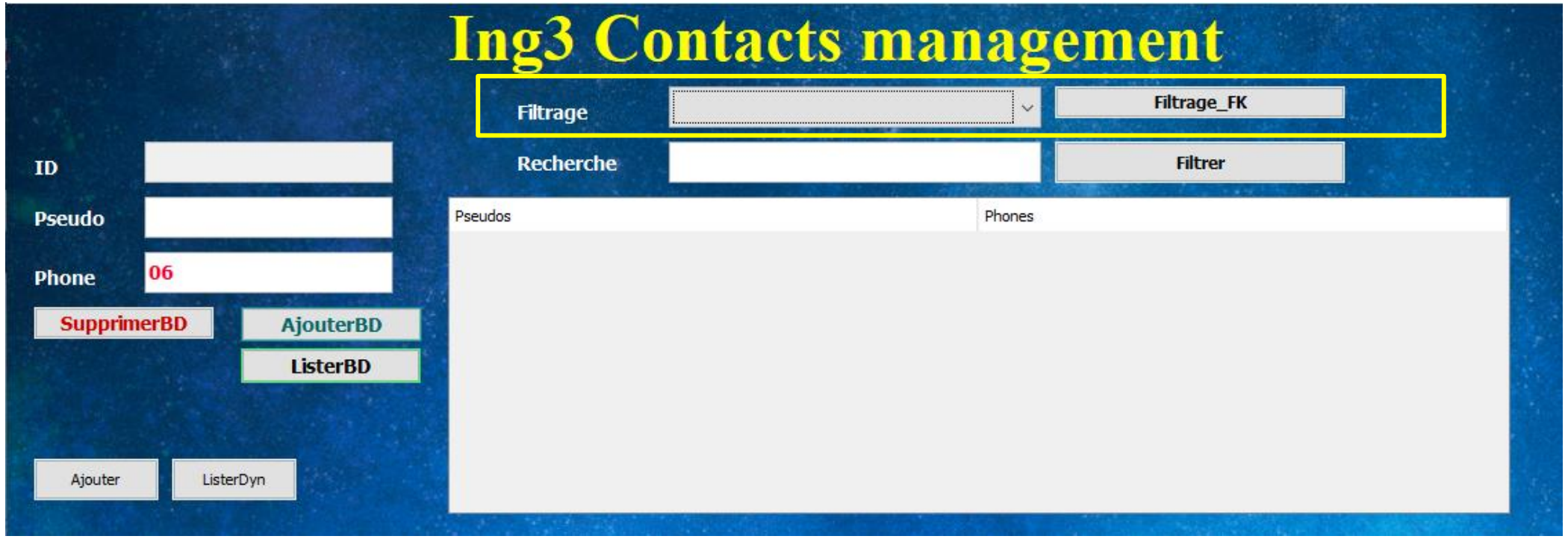
Exemple de filtrage dynamique

Factorisation du code pour rafraichir la jTable

```
public void refreshJtable (LinkedList<Contact> listeContact){

    Vector<String> VCols = new Vector<String>();
    VCols.add("ID");
    VCols.add("Pseudo ");
    VCols.add("Phone Number");

    // Creation des lignes temp
    Vector<Object> Vligne = new Vector<Object>();
    Vector<Vector<Object>> VData = new Vector<Vector<Object>>();

    for(Contact c:listeContact){
        // iteration sur les objests
        Vligne = new Vector<Object>();
        Vligne.add(c.id);
        Vligne.add(c.pseudo);
        Vligne.add(c.phone);
        // store in VData
        VData.add(Vligne);
    }
    DefaultTableModel modelDyn = new DefaultTableModel(VData, VCols
    jTable1.setModel(modelDyn);
}
```

Relation one to Many

Nous souhaitons enrichir notre exemple de gestion des contactes par l'intégration des catégories de contactes



Relation one to Many

Exemple de la base de données table catégorie

Nom	Type	Interclassement
id	int(11)	
pseudo	varchar(20)	utf8mb4_general_ci
phone	varchar(20)	utf8mb4_general_ci
idCateg	int(10)	

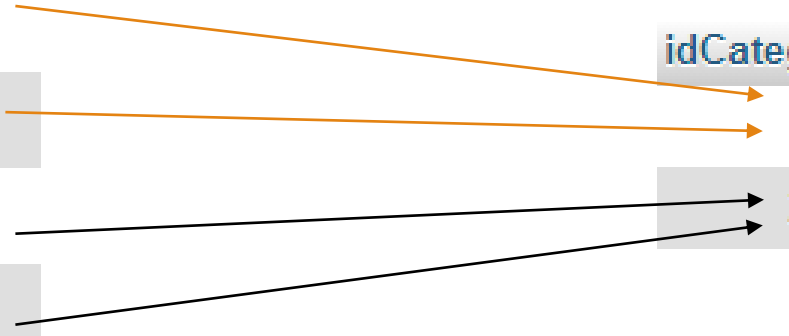
	#	Nom	Type	Interclassement
☐	1	idCateg	int(11)	
☐	2	nomCateg	varchar(20)	utf8mb4_general_ci
☐	3	DescCateg	varchar(150)	utf8mb4_general_ci

Relation one to Many

Vu sur la BD

id	pseudo	phone	idCateg
1	najib	06 064	1
2	Mehdi	06 67	1
4	nadir	06 7777	2
5	Reda	06 7777	2

idCateg	nomCateg	DescCateg
1	famille	famille
2	bros	homies and friends



Relation one to Many

- Remplissage dynamique de la Jcombox
- Event de chargement du Frame **formWindowOpened**

```
private void formWindowOpened(java.awt.event.WindowEvent evt) {
    LinkedList<Contact> listeContact = Contact.getAllContact();
    // refresh jtable
    this.refreshJtable(listeContact);
    // chargement de la JCBCategorie
    this.initJCBCategorie();
}
```

Initialisation JComboBox

Lister les categories

```
public static LinkedList<Categorie> getAllCategories() {
    LinkedList<Categorie> listeCateg= new LinkedList<Categorie>();
    try {
        java.sql.Connection conn = SingletonConnection.getConnection();
        String req = "SELECT * FROM categorie ";
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(req);

        Categorie c = null;
        while (rs.next()) {
            int id          = rs.getInt(1);
            String nom      = rs.getString(2);
            String description = rs.getString(3);

            c = new Categorie(id, nom, description);
            listeCateg.add(c);
        }
        rs.close();
        stmt.close();
    } catch (SQLException e) {
        System.out.println("Problème durant la récupération de la liste
                + "des categorie");
    }
    return listeCateg;
}
```

Initialisation JComboBox

Remplissage simple de la jcombobox

```
public void initJCBCategorie(){
    LinkedList<Categorie> lc = Categorie.getAllCategories();
    [redacted]

    for(int i = 0; i< lc.size(); i++){
        jcbCateg.addItem(lc.get(i).nomCateg);
        [redacted]
    }
}
```

Initialisation JComboBox

Gestion des indices et primary keys avec une HashTable

```
public void initJCBCategorie(){
    LinkedList<Categorie> lc = Categorie.getAllCategories();
    htCateg = new Hashtable();

    for(int i = 0; i< lc.size(); i++){
        jcbCateg.addItem(lc.get(i).nomCateg);
        // mise à jour du citionnaire
        htCateg.put(i, lc.get(i).idCateg);
    }
}
```

Initialisation JComboBox

Le choix de la Jtable est justifié par sa structuration du code

Hash Table <Key , Value>

- Key correspond à l'indice dans la JCOmboBox
- Value correspond à la primaryKey

Key index JCB	Values id dans BD
1 →	23
2 →	24
3 →	50

Pour ajouter un élément **hashtable.put**(clé , valeur)

Pour récupérer une valeur v = **hashtable.get**(clé)

Initialisation JComboBox

Exemple d'utilisation pour le filtrage par catégorie

```
private void jbtnFiltreCategActionPerformed(java.awt.event.ActionEvent evt) {
    int index = jcbCateg.getSelectedIndex();
    int realId = (int) htCateg.get(index);
    LinkedList<Contact> listeContact = Contact.filtrageByCateg(realId);
    System.out.println(listeContact.size());
    this.refreshJtable(listeContact);
}
```