

# Rapport des travaux pratiques

Filière : **Ingénierie des Systèmes d'Information et de Communication (ISIC)**

2<sup>e</sup> année Cycle Ingénieur

## JDBC:JavaDataBaseConnectivity



:

(Module : Programmation Réseau et Distribuées JAVAEE)

**Réalisé par :**

Errokbi oumaïma

**Encadré par :**

Dr. Mohamed LACHGAR

Année universitaire : 2023/2024

# Remerciements

*Je tiens à exprimer ma sincère gratitude à Dr. Mohamed Lachgar pour son encadrement, ses conseils précieux et son soutien constant tout au long de la réalisation de ce rapport. Sa passion pour le sujet et son expertise ont été une source d'inspiration et m'ont permis d'approfondir mes connaissances. Je lui suis reconnaissant pour sa disponibilité et sa patience, ainsi que pour l'orientation qu'il m'a fournie, qui ont grandement contribué à l'aboutissement de ce travail.*

# Résumé

*Ce rapport présente une exploration pratique de l'utilisation de JDBC avec le langage de programmation Java, en mettant un accent particulier sur l'exécution de requêtes SQL pour interagir avec une base de données. Nous allons plonger en profondeur dans les classes et interfaces fournies par le package `java.sql`, tout en exploitant SQL comme système de gestion de base de données (SGBD). En plus de l'intégration entre Java et les bases de données, nous aborderons également la notion d'encapsulation en Java, en démontrant comment structurer le code pour protéger et gérer efficacement les données. Cette approche vise à offrir une compréhension complète des concepts de programmation orientée objet en Java, tout en illustrant l'interaction avec les bases de données via des requêtes SQL.*

*Dans le **TPI**, nous travaillerons sur un exemple simple pour illustrer le processus de connexion 'un programme à une base de données.*

*Le **TAF (travail à faire)** approfondira les concepts abordés en explorant plus en détail le code et en proposant des méthodes de gestion, tout en utilisant des packages appropriés pour structurer et optimiser le développement.*

# TP1 : Insertion et récupération des données

---

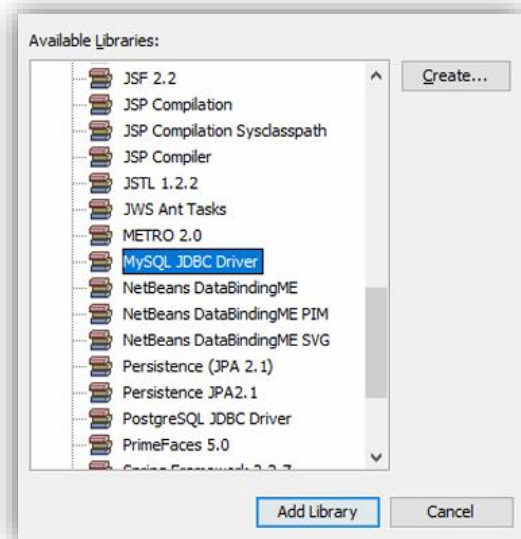
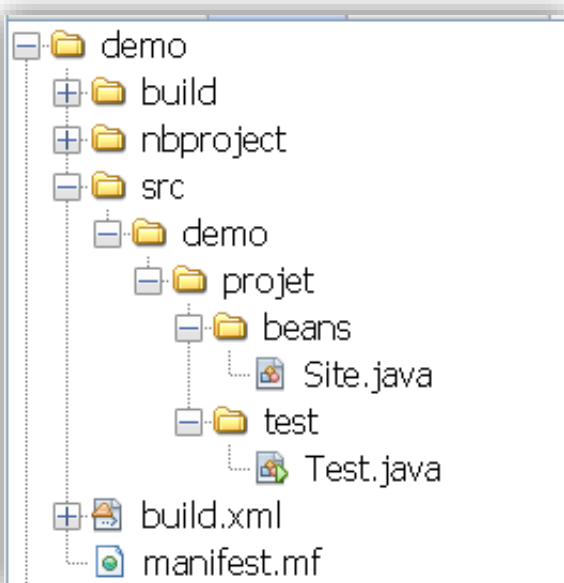
## 1.1 But de TP :

Maîtrisez l'insertion et la récupération de données dans une base de données MySQL ou ORACLE avec JAVA. Manipulez les objets Connexion, Statement, ResultSet. Gérez les erreurs potentielles.

## 1.2 Etapes:

- **Partie 1** : *projet netbeans*.

Avant de commencer il est nécessaire de structurer le projet on utilise des packages qui



encapsule les classes utilise.

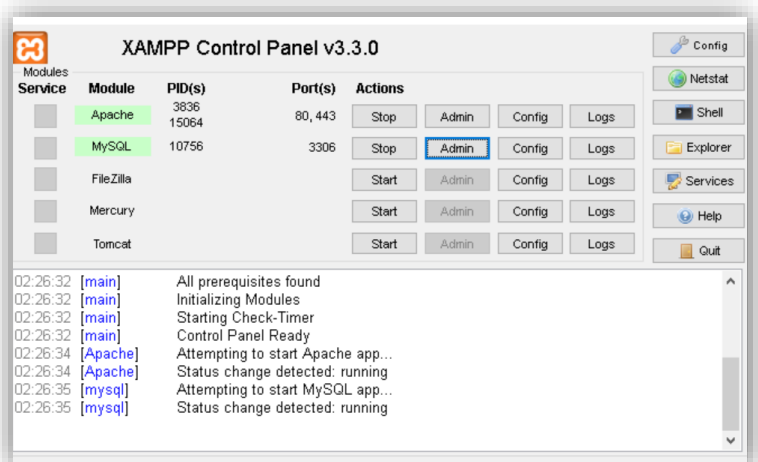
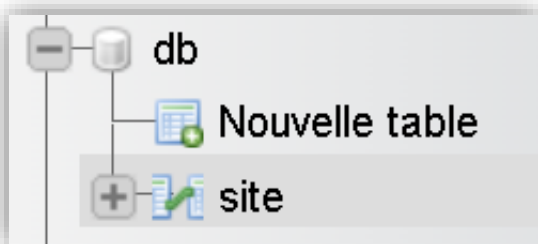
Sans oublier de télécharger le driver *MYSQL JDBC DRIVER* :

Commençons tout d'abord par le package beans ou on trouve la classe Site qui présente l'entité site.

- **Partie 2:** *Création de la table Site.*

Utilisation de SJBD dans notre cas MYSQL afin de crée une table site dans la base de donnee nomme 'db', utilisation dans ce cas de XAMP, *PHPMYADMIN*.

On utilise les requêtes SQL nous créons la base de donnée db ainsi que la table site.

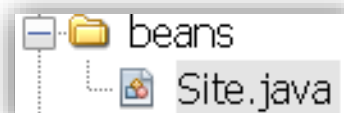


```
1 CREATE DATABASE db;
2 USE db;---- Structure de la table `site`-
3 CREATE TABLE `site` (
4   `id` int(11) primary key auto_increment,
5   `nom` varchar(100) NOT NULL
6 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- **Partie 3:** *insertion des données*

Créer la methode Site dans le package beans :

```
22 public class Test {
23     public static void save(Site s){
24         String url ="jdbc:mysql://localhost:3306/db";
25         String password="";
26         String user ="root";
27         Connection cn=null;
28         Statement st =null;
29         try {
30             Class.forName("com.mysql.jdbc.Driver");
31             cn = DriverManager.getConnection(url, user, password);
32             String req= "INSERT INTO `site` VALUES (null,'"+ s.getNom() +"')";
33             st=cn.createStatement();
34             int n =st.executeUpdate(req);
35         }
36         catch(SQLException e ){
37             Logger.getLogger(Connection.class.getName()).log(Level.SEVERE, null, e);
38         }
39     }
40     catch(ClassNotFoundException ex ){
41         System.out.println("impossible de charger le driver ");
42     }
43 }
44 finally {
45     try {
46         st.close();
47         cn.close();
48     }
49     catch(SQLException e) {
```

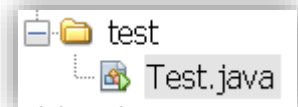


d'où on va avoir une méthode d'insertion .

- **Partie 4: Test**

Maintenant on va créer un package Test dont lequel on va insérer la classe Test pour effectuer le Test

```
public static void save(Site s) {
    String url = "jdbc:mysql://localhost:3306/db";
    String password = "";
    String user = "root";
    Connection cn = null;
    Statement st = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        cn = DriverManager.getConnection(url, user, password);
        String req = "INSERT INTO `site` VALUES (null, '"+ s.getNom() + "')";
        st = cn.createStatement();
        int n = st.executeUpdate(req);
    }
    catch (SQLException e) {
        Logger.getLogger(Connection.class.getName()).log(Level.SEVERE, null, e);
    }
    catch (ClassNotFoundException ex) {
        System.out.println("impossible de charger le driver ");
    }
    finally {
        try {
            st.close();
            cn.close();
        }
        catch (SQLException e) {
            System.out.println("impossible de liberer les ressources ");
        }
    }
}
```




Création de la méthode save (Site s) dans la classe test :

Insérer des données :

```
88 public static void main(String[] args) {
89     save(new Site("SAFI"));
90     save(new Site("MARRAKECH"));
91     save(new Site ("EL JADIDA"));
92     load();
93 }
```

Consulter la base de données db pour savoir si les données sont insérées ou pas :

						id	nom	
<input type="checkbox"/>		Éditer		Copier		Supprimer	1	SAFI
<input type="checkbox"/>		Éditer		Copier		Supprimer	2	MARRAKECH
<input type="checkbox"/>		Éditer		Copier		Supprimer	3	EL JADIDA

- **Partie 5:** Insertion de donne

Dans la classe Test on va une méthode load () permettant d'afficher les différents sites :

```

55 public static void load() {
56     String url = "jdbc:mysql://localhost:3306/db";
57     String user = "root";
58     String password = "";
59     Statement st = null;
60     Connection cn = null;
61     try {
62         Class.forName("com.mysql.jdbc.Driver");
63         cn=DriverManager.getConnection(url, user, password);
64         String req= " Select * from Site ";
65         st =cn.createStatement();
66         ResultSet n = st.executeQuery(req);
67         while (n.next()) {
68             System.out.println(n.getInt(1)+" "+ n.getString(2));
69         }
70     }
71     } catch (ClassNotFoundException e) {
72         System.out.println("Driver not found ");
73     } catch (SQLException ex) {
74         System.out.println(ex.getMessage());
75     } finally {
76         try {
77             st.close();
78             cn.close();
79         } catch (SQLException ex) {
80

```










- **Partie 6:** Insertion de donne

```

88 public static void main(String[] args) {
89     //save(new Site("SAFI"));
90     //save(new Site("MARRAKECH"));
91     //save(new Site ("EL JADIDA"));
92     load();
93 }

```

Consulter la base de donne pour voir si l'information son insérer ou pas :

					id	nom
<input type="checkbox"/>		Éditer		Copier		Supprimer
					1	SAFI
<input type="checkbox"/>		Éditer		Copier		Supprimer
					2	MARRAKECH
<input type="checkbox"/>		Éditer		Copier		Supprimer
					3	EL JADIDA

## **1.3 Conclusion :**

On utilise l'API JDBC ainsi que le package `java.sql` on a pu créer des méthodes dans des classes afin de tester le processus d'insertion et de récupération des données à partir d'une base de données comme `db` dans notre cas. Soulignons aussi l'importance des classes et des interfaces du package déjà mentionné.



# Travail à rendre : script des développeurs

---

## 1.1 But de TP :

Le but de ce rapport est de suivre l'avancement du travail des développeurs au sein d'un projet, en se concentrant sur la réalisation de scripts. Un rapport hebdomadaire, établi par le chef de projet, recense quotidiennement le nombre de scripts réalisés par chaque développeur. Ces données sont collectées et stockées dans un système de gestion de base de données (SGBD) afin de faciliter le suivi et l'analyse de la progression du projet.

## 1.2 Etapes:

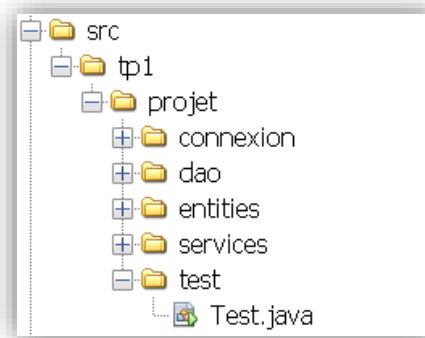
**Exercice 1 :** *Ouvrir une connexion, Créer et remplir une table.*

- **Partie 1 :** *création de projet*
  - ouverture d'une connexion à la base MySQL;
  - création de la table récapitulative des réalisations des scripts;
  - initialisation de la table;
  - supprimer la table;
  - compiler et tester votre programme

Avant de commencer il' est nécessaire de savoir la structure du projet :

Chaque package présente des classe spécifique qu'on va les mentionner apres.

Sans oublier de telecharger le driver *MYSQL JDBC DRIVER* :



Chaque **package** est présentée comme suit :

- package connexion : va contenir une class connexion qui va faire liee la connexion avec la base de donnee.
- package dao : va contenir une interface générique qui contrôle le comportement du service de chaque entité.
- package entité : contient les entités utilise dans le projet.
- package service : vas contenir les services que cette application offre, les classes présente dans ce package son implémente d'interface dao.
- package Test : ou on va effectuer les test.

## • Partie 2 : Création de la table

Ona le choix d'utiliser deux methode soit de créé la table au niveau de phpmyAdmin ou bien utiliser des méthodes au niveau de code java.

Au niveau du package ExoJDBC on va créer une classe qui implémente l'interface générique Idao dont laquelle on va créer deux methode :

- **creatDtable ()** : permet de créé une table au niveau de la base de donne.
- **update (DevData o)** : permet d'insérer des perssone au niveau de la table.

Voici les méthodes :

```

public Boolean update(DevData o) {
    Boolean r= false;
    try {
        String req ="INSERT INTO devdata (developpeurs , jour, NBScripts) VALUES ('"
            + o.getDeveloppeur() + ", '"
            + o.getJour() + "', "
            + o.getNbScript() + ")";
        Statement st = Connexion.getCn().createStatement();
        int n = st.executeUpdate(req);
        if (n==1) {
            r=true ;
        }
    } catch( SQLException e ){
        Logger.getLogger(DevData.class.getName()).log(Level.SEVERE, null, e);
    }
    return r;
}

```

```

@Override
public void creatDtable() {
    try {
        String req ="CREATE TABLE devdata (" +
            "developpeurs VARCHAR(32), " +
            "jour CHAR(11), " +
            "NBScripts INTEGER)";

        Statement st = Connexion.getCn().createStatement();
        int n= st.executeUpdate(req);
        if (n==1) {
            System.out.println("la table est cree");
        } else {
            System.out.println("table non cree ");
        }
    } catch( SQLException e) {
        Logger.getLogger(DevData.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

Au niveau de la classe Test dans le package test on va effectuer ces méthodes pour créer notre table et effectuer aussi l'insertion des donnee. (en commentaire car j fait run une seul fois pour la création et l'insertion s'effectue une seule fois)

```
ExoJDBC e1 = new ExoJDBC();
//e1.creatDtable();
//DevData d1= new DevData("ALAMI", "LUNDI", 1);
//DevData d2= new DevData("WAFI", "LUNDI", 2);
// DevData d3= new DevData("SALAMI", "MARDI", 9);
// DevData d4= new DevData("SAFI", "MARDI", 2);
// DevData d5= new DevData("ALAMI", "MARDI", 2);
// DevData d6= new DevData("SEBIHI", "MERCREDI", 2);
// DevData d7= new DevData("WAFI", "JEUDI", 3);
// DevData d8= new DevData("ALAOUI", "VENDREDI", 9);
// DevData d9= new DevData("WAFI", "VENDREDI", 3);
// DevData d10= new DevData("SEBIHI", "VENDREDI", 4);
// e1.update(d1);
// e1.update(d2);
// e1.update(d3);
//e1.update(d4);
//e1.update(d5);
//e1.update(d6);
//e1.update(d7);
//e1.update(d8);
//e1.update(d9);
//e1.update(d10);
```

Consultation de la base de donne :

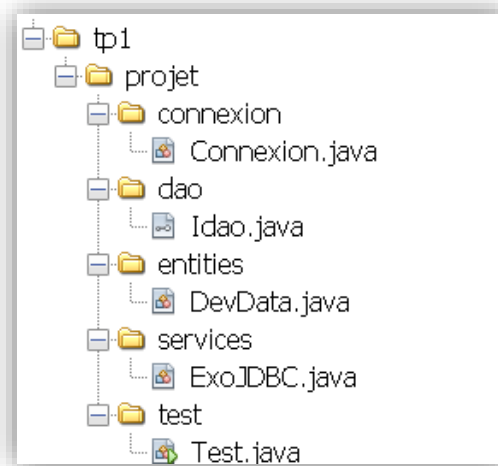
developpeurs	jour	NBScripts
ALAMI	LUNDI	1
WAFI	LUNDI	2
SALAMI	MARDI	9
SAFI	MARDI	2
ALAMI	MARDI	2
SEBIHI	MERCREDI	2
WAFI	JEUDI	3
ALAOUI	VENDREDI	9
WAFI	VENDREDI	3
SEBIHI	VENDREDI	4

✓ Opération effectué

**Exercice 2 :** *Rechercher de l'information dans la base de données*

- **Partie 1 :** *création des services*

Création de la classe ExoJDBC qui implemente l'interface Idao qui offre les services nécessaire pour l'application utilise



Création de l'interface Idao :

Chaque methode permet d'effectuer une opération donne :

- **nbrmaxS ()** : la personne ayant réalisé le nombre maximum de script en une journée :
- **requette ()** : le nombre de colonnes de la table résultat est affiché;pour chaque colonne, son nom et le type des données est affiché;le contenu de la table est affiché ligne par ligne. Sinon le nombre de lignes modifiées dans la table DevData est affiché .
- **trieDec ()**:la liste des personnes triée dans l'ordre décroissant selon leur nombre de scripts.
- **nbrtotalsS ()**: calculer et afficher le nombre total de scripts réalisés en une semaine.

```
public interface Idao <T>{
    List<T> nbrmaxS ();
    void requette ();
    void trieDec ();
    void nbrtotalsS ( );
    Boolean update (T o);
    void creatDtable ( );
}
```

- **update (T o)**:d'insérer des donnee dans la table
- **creatDtable ()** : créé une table donnee.

Bon on va expliquer le script de chaque methode :

## nbrmaxS () :

```
public List<DevData> nbrmaxS() {
    List<DevData> Data = new ArrayList<>();
    try {
        Statement st = Connexion.getCn().createStatement();
        String req = "SELECT Developpeurs, jour, MAX(NBScripts) FROM devdata GROUP BY jour";
        ResultSet sn = st.executeQuery(req);
        while (sn.next()) {
            Data.add(new DevData(sn.getString("developpeurs"), sn.getString("jour"), sn.getInt(3)));
        }
    } catch (SQLException ex) {
        Logger.getLogger(DevData.class.getName()).log(Level.SEVERE, null, ex);
    }
    return Data;
}
```

## trieDec ():

```
public void trieDec() {
    List<DevData> Datas = new ArrayList<>();
    try {
        Statement st = Connexion.getCn().createStatement();
        String req = "SELECT Developpeurs, sum(NBScripts) as c FROM devdata GROUP BY Developpeurs order by c desc ";
        ResultSet sn = st.executeQuery(req);
        while (sn.next()) {
            Datas.add(new DevData(sn.getString("developpeurs"), null, sn.getInt(2)));
        }
    } catch (SQLException e) {
        Logger.getLogger(DevData.class.getName()).log(Level.SEVERE, null, e);
    }
    System.out.println("Liste des personnes lisee par ordre decroissant selon leur nombre de script : ");
    for (DevData e : Datas) {
        System.out.println("-nom : " + e.getDeveloppeur() + " ; le nombre de script : " + e.getNbScript() + ".");
    }
}
```

## nbrtotalS ():

```
public void nbrtotalS() {
    boolean r = false;
    try {
        Statement st = Connexion.getCn().createStatement();
        String req = "Select SUM(NBScripts) as c from devdata";
        ResultSet sn = st.executeQuery(req);
        if (sn.next()) {
            r = true;
            System.out.println(" nombre totale des Script realisee en une semaine : " + sn.getInt(1));
        }
    } catch (SQLException e) {
        Logger.getLogger(DevData.class.getName()).log(Level.SEVERE, null, e);
    }
}
```

Et on a d'autres méthodes supplémentaires qui n'implémentent pas de classe Dao.

**nbrtotalSJ ()** : calculer pour un programmeur donné le nombre total de scripts réalisés

```
public void nbrtotalSJ(String dt) {
    boolean r = false;
    try {
        Statement st = Connexion.getCn().createStatement();
        String req = "Select Developpeurs ,SUM(NBScripts) from devdata GROUP BY Developpeurs ";
        ResultSet sn = st.executeQuery(req);
        while (sn.next()) {
            if (dt.trim().equalsIgnoreCase(sn.getString("developpeurs").trim())) { // trim() pour enlever les espaces et en utilisant
                System.out.println(" nombre totale des Script realisee par " + sn.getString(1) + " est : " + sn.getInt(2));
                r = true;
            } else {
                System.out.println("ce nom n'existe pas dans la base de donnee");
            }
            break;
        }
    } catch (SQLException ex) {
        Logger.getLogger(ExoJDBC.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

**Exercice 2 :** *Effectuer une requête libre et obtenir la méta information sur les types de données du résultat*

- **Partie 1 :** *création d'une methode de la requette libre*

Cree une methode qui permet de :

- le nombre de colonnes de la table résultat est affiché.
- pour chaque colonne, son nom et le type des données est affiché.
- le contenu de la table est affiché ligne par ligne. Sinon le nombre de lignes modifiées dans la table DevData est affiché.

Donc on va créer une methode requette () qui va donner l'opportunité a l'utilisateur d'entrer une requette, ainsi que donner des informations plus détailler sur cette requette

```
public void requette () {  
    boolean r=false;  
    System.out.println("entrer la requette : ");  
    Scanner sc = new Scanner(System.in);  
    try {  
        Statement st =Connexion.getCn().createStatement() ;  
        String req= sc.nextLine();  
        ResultSet sn =st.executeQuery(req);  
        ResultSetMetaData data = sn.getMetaData();  
        System.out.println("information sur la requette fournit ");  
        int c= data.getColumnCount();  
        System.out.println("le nombre de colonne de la table est :"+ data.getColumnCount());  
        System.out.println("");  
        if (req.trim().toLowerCase().startsWith("select")) {  
            r=true;  
  
            for (int i = 1; i <= c; i++) {  
                String columnName = data.getColumnName(i);  
                String columnType = data.getColumnTypeName(i);  
                System.out.println("Colonne " + i + " : " + columnName + " (Type: " + columnType + ")");  
            }  
  
            while (sn.next()) {  
                for (int i = 1; i <= c; i++) {  
                    System.out.print(sn.getString(i) + " ");  
                }  
                System.out.println();  
            }  
        }  
    }  
}
```

- **Partie 2 :** *Test Global*

On va passer au package test au niveau de la class  
test on va effectuer un test de l'ensemble des  
methode effectuée de l'ensemble des méthodes au  
niveau de main

```
for(DevData e : e1.nbrmaxS()) {  
    System.out.println(e);  
}  
  
e1.trieDec();  
e1.nbrtotalS( );  
e1.nbrtotalSJ("ala");  
e1.requette ( );  
}
```

Ala ne se trouve pas dans la base de donne :

Donc on obtient le resultat suivante :

ce nom n'existe pas dans la base de donnee

C'est c qu'on veut de la part de la methode

Ainsi que les autre methode :

```
Output - TP1 (run) x  
DevData{developpeur=SALAMI, jour=MARDI, nbScript=9}  
DevData{developpeur=SEBIHI, jour=MERCREDI, nbScript=2}  
DevData{developpeur=ALAOUI, jour=VENDREDI, nbScript=9}  
*liste des perssones liee par ordre decroissant selon leur nombre de  
-nom : ALAOUI ; le nombre de script : 9.  
-nom : SALAMI ; le nombre de script : 9.  
-nom : WAFI ; le nombre de script : 8.  
-nom : SEBIHI ; le nombre de script : 6.  
-nom : ALAMI ; le nombre de script : 3.  
-nom : SAFI ; le nombre de script : 2.  
  nombre totale des Script realisee en une semaine : 37  
ce nom n'existe pas dans la base de donnee  
entrer la requette :
```

✓ Opération effectue

## 1.3 Conclusion :

On utilise l'API JDBC ainsi que le package `java.sql` on a pu créer des méthodes dans des classes afin de tester le processus d'insertion et de récupération des données à partir d'une base de données comme `db` dans notre cas. Soulignons aussi l'importance des classes et des interfaces du package déjà mentionné.



