Université Cadi Ayyad
Ecole Supérieure de Technologie – Safi

COURS 7

COURS DE LA PROGRAMMATION ORIENTÉE OBJET (JAVA)

Pr. EL ABDELLAOUI SAID

Elabdellaoui.said@yahoo.fr

JDBC

PLAN

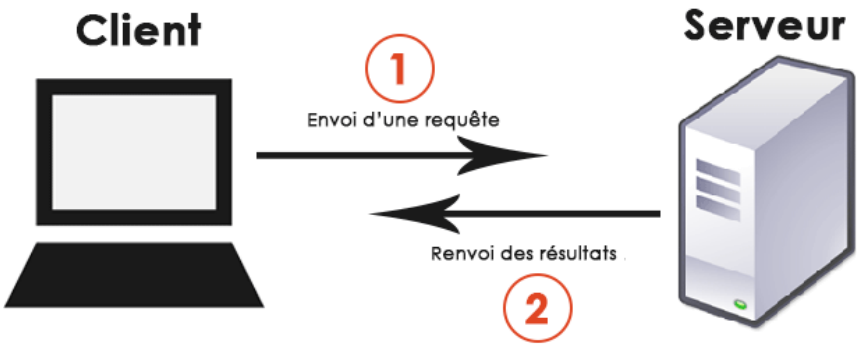
- **1** : Présentation
- **2** : Java DataBase Connectivity(JDBC)
- **3** : Connexion à une base de données
- **4** : Les requêtes de sélection
- **5** : Les requêtes de mise à jour
- **6** : Déconnexion
- **7** : SQLException
- **8** : Instruction SQL paramétrée

PRÉSENTATION

JAVA DATABASE CONNECTIVITY (JDBC)

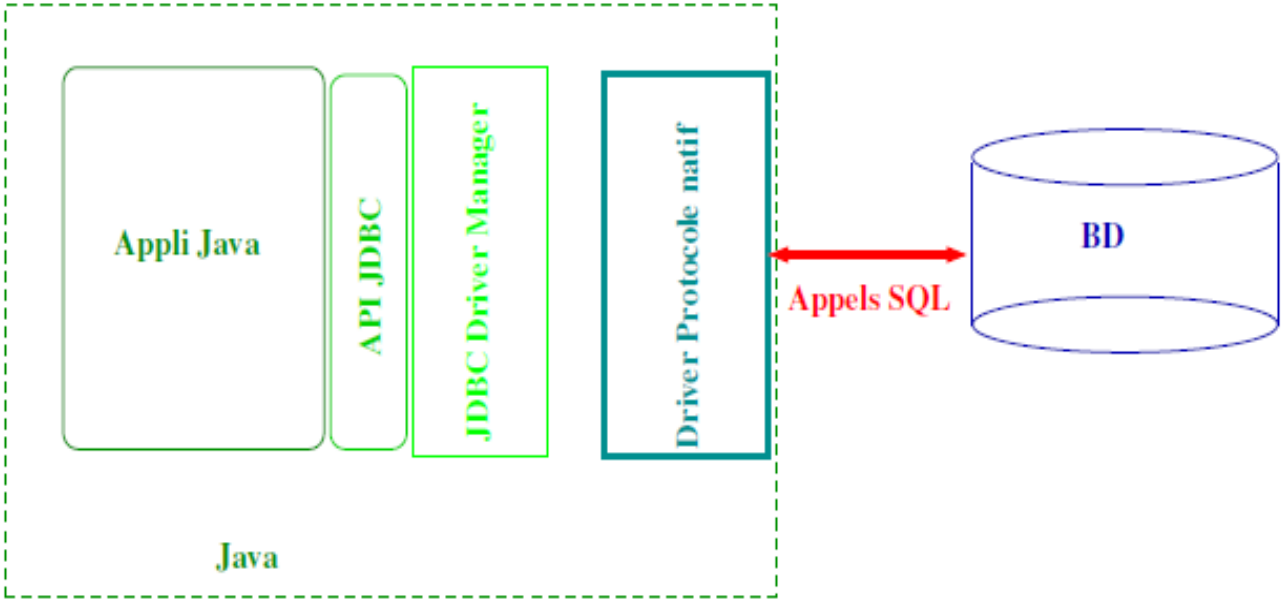
- **JDBC** (Java DataBase Connectivity): c'est la relation entre Java et les bases de données.
- **JDBC** : un driver (*pilote*) fournit des outils pour traiter les *BD*.
- **JDBC** permet à un programme java d'interagir localement ou à distance avec une base de données relationnelle.
- Fonctionne selon le principe Client-Serveur

Programme JAVA

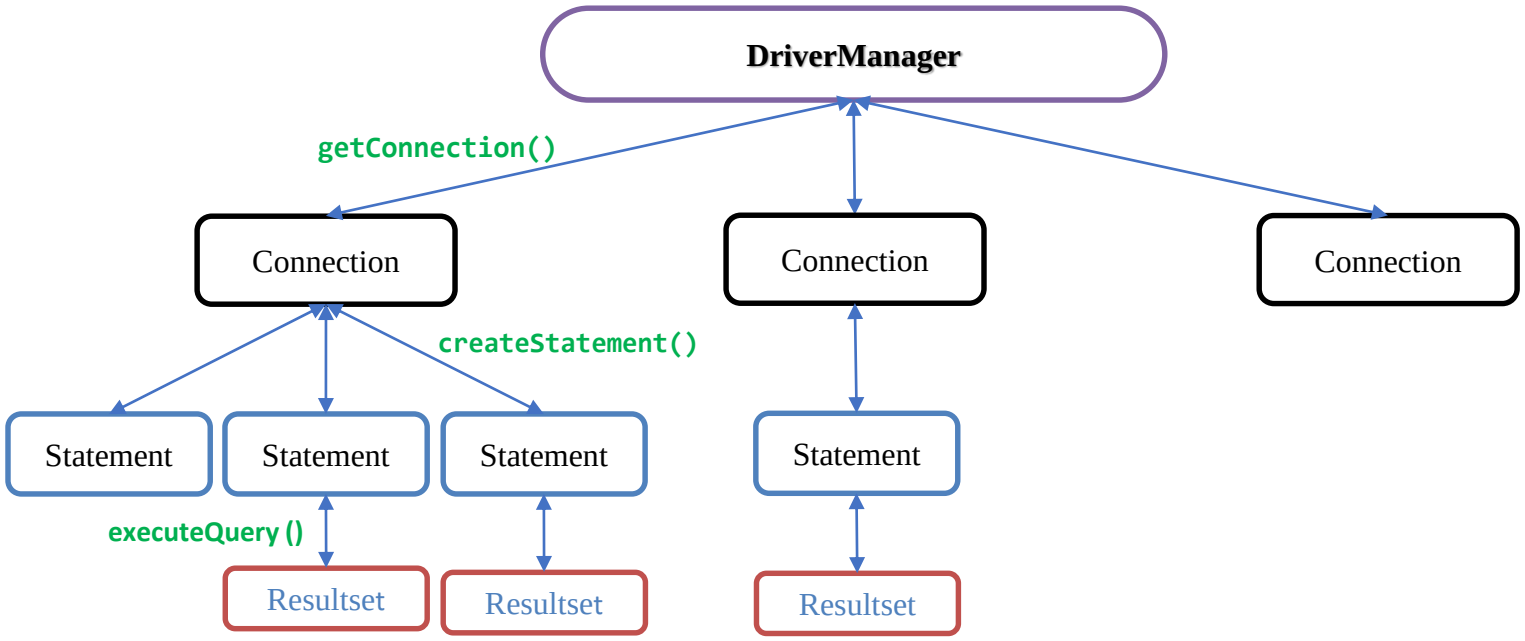


SGBD (ex :phpMyAdmin,MySQL…)

GÉNÉRALITÉS



GÉNÉRALITÉS



Canal de communication TCP vers une BD

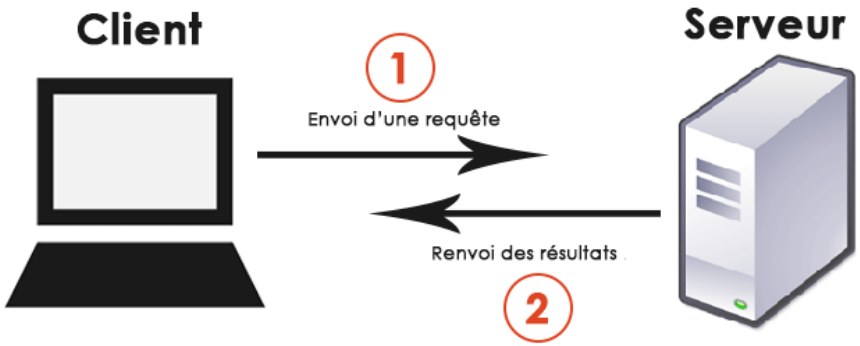
Gérer les requêtes SQL

Gère l'accès aux tuples d'un résultat

JAVA DATABASE CONNECTIVITY (JDBC)

- **JDBC** permet à un programme java d'interagir localement ou à distance avec une base de données relationnelle.
- Fonctionne selon le principe Client-Serveur

Programme JAVA



SGBD (ex :phpMyAdmin,MySQL…)

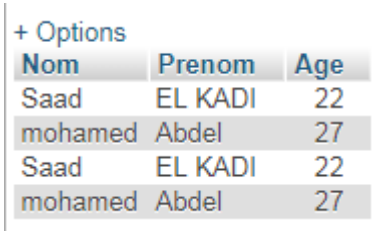
- Les **3** opération suivantes doivent être réalisées :

1. Installer un server de *BD* : *phpMyAdmin, MySQL etc* .
2. Charger le diver *JDBC*.
3. Etablir la connexion avec la base de donnés pour l'interroger.

OPERATIONS

Opération 1 :

- Télécharger *phpMyAdmin* du site officiel de *phpMyAdmin*
- Installer *phpMyAdmin*
- Utiliser un serveur de base de données de votre choix : *XAMPP, MySQL ,oracle…ect*
- **Préparation :**
  - Créer une base de donnée nommé «BDTest» contenant une table nommée «Personne». Cette table contient 3 champs : Nom(Varchar(10)), Prenon(Varchar(10)) et Age(int)).
  - Remplissez

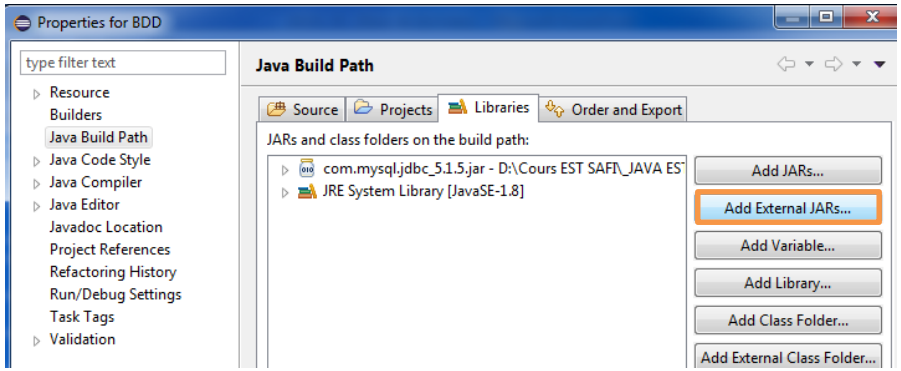


Opération 2 :

- Télécharger le driver **JDBC** nommé "*mysql-connector-java*" du site officiel de *MySQL* . C'est un dossier compressé. Après décompression ,vous allez trouver le fichier *jar* correspondant au driver.
- Ajouter le driver **JDBC** à votre projet comme suit :

  Cliquer avec le bouton droit de la souris sur le nom du projet dans l'explorateur

  Eclipse -> Propriétés->Java Build Path -> onglet *Librairies*->cliquer sur le bouton ''*add external JAR*"->sélectionner le *driver JDBC* de *Mysql*.



Opération 3:

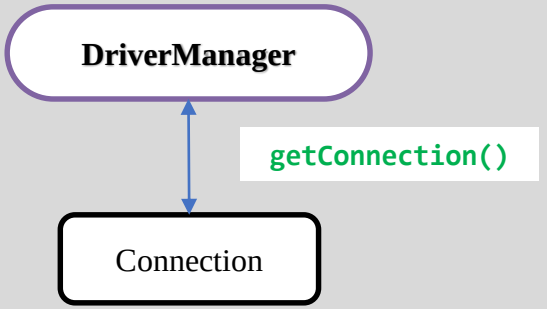
- Ouvrir le projet auquel nous avons ajouter le driver.
- Ajouter des classes au projets et via des instruction Java,
- Le programme peut:

  - ✓ **Charger** le driver JDBC
  - ✓ **Etablir** la connexion avec la BD
  - ✓ **Exécuter** des requêtes SQL
  - ✓ **Traiter** les données retrouvées
  - ✓ **Effectuer** des traitements des données
  - ✓ **Fermer** la connexion

ARCHITECTURE D'UNE APPLICATION JDBC

- Architecture globale d'une application JDBC :

```
import java.sql.DriverManager;    //gestion des pilotes
import java.sql.Connection;      //une connexion à la BD
import java.sql.Statement;      // requêtes
import java.sql.ResultSet;     //un résultat (lignes / colonnes)
import java.sql.SQLException; // Erreur

public class BaseSQL {
public BaseSQL() {
try{
//Chargement pilote
//Ouverture de connexion
//Exécution d'une requête
//Traitement des résultats
//Traitements des données
//Fermeture de la connexion

} catch (Exception ex){   }
}
```



ARCHITECTURE D'UNE APPLICATION JDBC

- **Chargement** pilote **JDBC** :

(1)
```
Class.forName("com.mysql.jdbc.Driver");
```

- **Ouverture** d'une connexion avec la base de données :
  - On fournit l'url de la base , le nom d'utilisateur et le mot de passe.

(2)
```
String url="jdbc :SousProtocole :SourceDeDonnées";
Connection con=DriverManager.getConnection(url,"Utilisateur","MotDePasse");
```

  - Exemple :

```
String url="jdbc:mysql://localhost/BDTest";
con= DriverManager.getConnection(url,"root","");
```

EXEMPLE

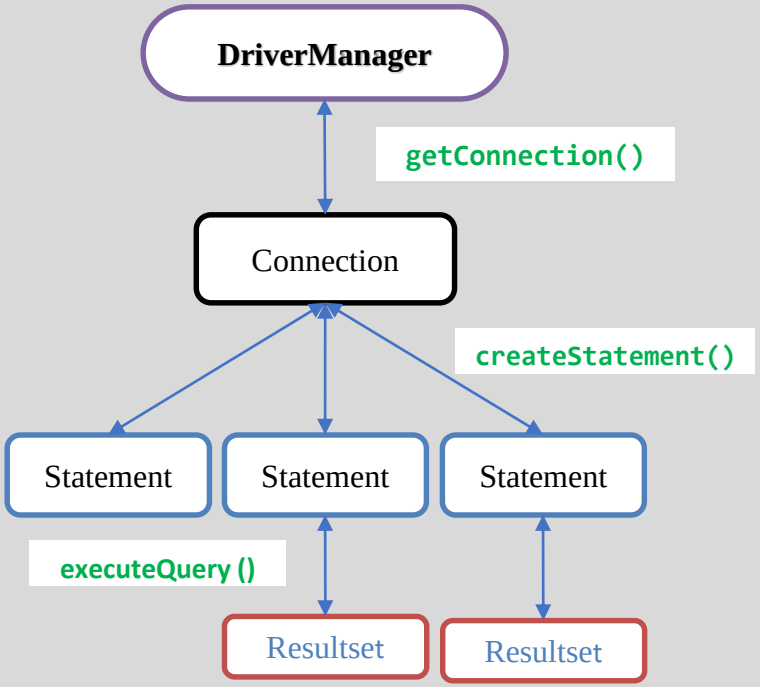
Tester !!
Si la connexion est bien rétablie !!"

```
import java.sql.DriverManager;    //gestion des pilotes
import java.sql.Connection;      //une connexion à la BD
import java.sql.Statement;      // requêtes

public class BaseSQL  {
     private Connection con;

  public BaseSQL() {
     try{
     //Chargement pilote
       Class.forName("com.mysql.jdbc.Driver");
     //Ouverture de connexion
       String url="jdbc:mysql://localhost/BDTest";
       con= DriverManager.getConnection(url,"root","");
        System.out.println("connexion bien rétablie !!");
      //Exécution d'une requête
     //Traitements des données
     //Fermeture de la connexion
       } catch (Exception ex){ System.out.println("Erreur :"+ex);
                    }
  public static void main(String [] args ) {new BaseSQL ();}
                     }
```

ARCHITECTURE D'UNE APPLICATION JDBC

- Architecture globale d'une application JDBC :

```
import java.sql.DriverManager;    //gestion des pilotes
import java.sql.Connection;      //une connexion à la BD
import java.sql.Statement;      // requêtes
import java.sql.ResultSet;     //un résultat (lignes / colonnes)
import java.sql.SQLException; // Erreur

public class BaseSQL {
public BaseSQL() {
try{
//Chargement pilote
//Ouverture de connexion
//Exécution d'une requête
//Traitement des résultats
//Traitements des données
//Fermeture de la connexion

} catch (Exception ex){   }
}
```



ARCHITECTURE D'UNE APPLICATION JDBC

- **Exécuter** d'une requête :
  - Pour réaliser des requêtes, on utilise un objet ***st*** de type **Statement**.

(3)
```
st =con.createStatement();
```

  - Le résultat d'une requête est récupéré par un objet ***res*** de type interface **ResultSet** et permet d'accéder aux données extraites grâce à la requête.

(4)
```
ResultSet res = st.executeQuery("select * from tab") ;
```

- **Traiter** les résultats :[2]
  - Après la requête, le "curseur" est positionné juste avant la première ligne du résultat.
  - La méthode *next()* permet d'avancer d'enregistrement en enregistrement séquentiellement :

(5)
```
res.next();
```

  - Pour récupérer les données dans chaque colonne, l'interface **ResultSet** propose plusieurs méthodes adaptées aux types des données récupérées :

```
getString(NumCol),getInt(NumCol),getDate(NumCol).
```

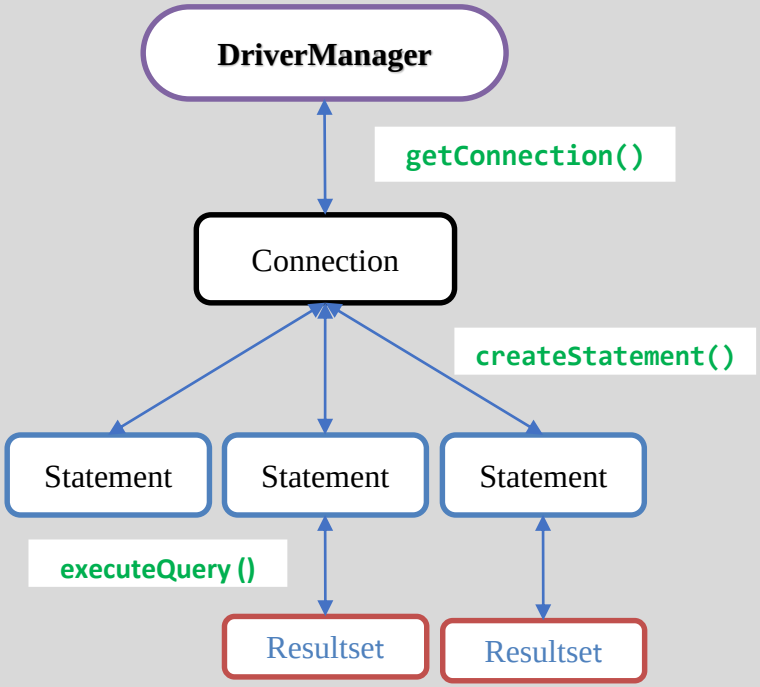
EXEMPLE


```
//……
import java.sql.ResultSet;
public class BaseSQL  {
     //……
     private ResultSet rs;
      private Statement st;
public BaseSQL() {
     try{   //……
     //Exécuter d'une requête
      st =con.createStatement();
      String requete = "select * from Personne";
      rs = st.executeQuery(requete);
     //Traiter les résultats (affichage)
      System.out.println("Enregistrements de la table");
      while(rs.next()){
        String Nom = rs.getString("nom");
        String Prenom = rs.getString("prenom");
        int Age = rs.getInt("age");
        System.out.println( "Nom: "+Nom+" Prenom: "+Prenom+ "Age: "+Age);
                    }
     //Fermeture de la connexion
       } catch (Exception ex){//……                }
                         }
```

ARCHITECTURE D'UNE APPLICATION JDBC

- Architecture globale d'une application JDBC :

```
import java.sql.DriverManager;    //gestion des pilotes
import java.sql.Connection;      //une connexion à la BD
import java.sql.Statement;      // requêtes
import java.sql.ResultSet;     //un résultat (lignes / colonnes)
import java.sql.SQLException; // Erreur
public class BaseSQL {
public BaseSQL() {
try{
//Chargement pilote
//Ouverture de connexion
//Exécution d'une requête
//Traitements des résultats
//Traitements des données
//Fermeture de la connexion

} catch (Exception ex){   }

}
```



EXEMPLE

❑ **Fermeture** de la connexion :

- La méthode *close()* permet de libérer les ressources prises par la création d'objets de type *ResultSet* , *Statement* et *Connection*.

(6)

```
con.close();
```

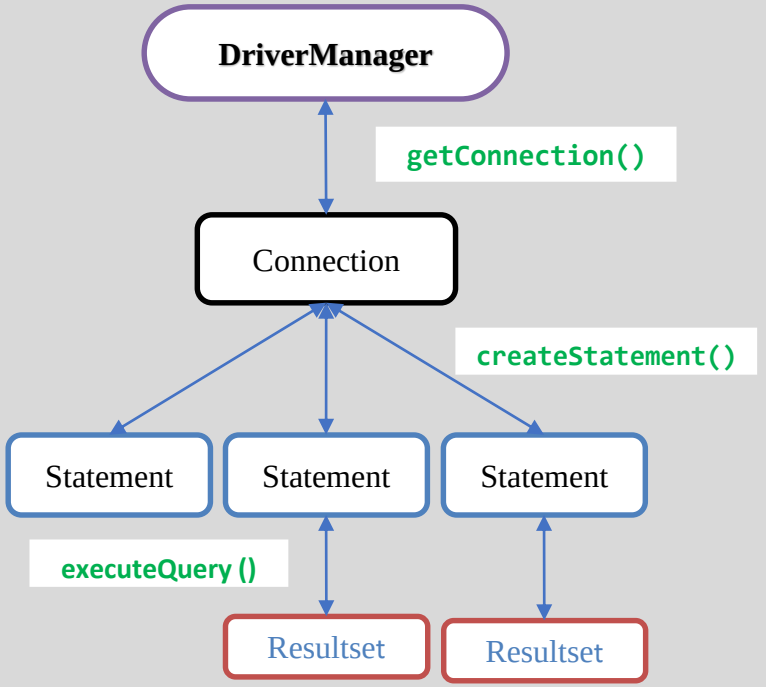
- Elle existe pour chacune de ces interfaces. Elle est le plus souvent employé pour une *Connection*, car elle libère en même temps toutes les ressources qui lui sont associées.

```
//……
import java.sql.ResultSet;

public class BaseSQL  {
     //……
public BaseSQL() {
     try{   //……
     //Fermeture de la connexion
     if(con != null){
        con.close();
     }

       } catch (Exception ex){//……                }
                          }
```

ARCHITECTURE D'UNE APPLICATION JDBC

- Architecture globale d'une application JDBC :

```
import java.sql.DriverManager;    //gestion des pilotes
import java.sql.Connection;      //une connexion à la BD
import java.sql.Statement;      // requêtes
import java.sql.ResultSet;     //un résultat (lignes / colonnes)
import java.sql.SQLException; // Erreur

public class BaseSQL {
public BaseSQL() {
try{
//Chargement pilote
//Ouverture de connexion
//Exécution d'une requête
//Traitements des résultats
//Traitements des données
//Fermeture de la connexion

} catch (Exception ex){   }
}
```



LES REQUÊTES DE MISES À JOUR

- La mise à jour d'une base de données peut être effectuée par une requête *SQL* de type :
  - ***INSERT***,
  - ***UPDATE*** ,
  - ***DELETE***

  à travers la méthoed ***executeUpdate("Requête")*** sur un objet de type *Statement*.
- Le résultat renvoyé par l'exécution de la requête indiquera le nombre de lignes mises à jour dans la base, contrairement à une requête de sélection qui renvoie un *ResultSet*.

LES REQUÊTES DE MISES À JOUR

- **Ajout** d'un enregistrement :

```
import java.sql.SQLException;
//……
public class BaseSQL {
//……
private Statement st;
//String N="", P=""; int A;

public BaseSQL () {
try {
//……
String requete1="INSERT INTO Personne VALUES('saber','Mohammed',20)";
// pour ajouter des valeurs non déterminées:
String requete="INSERT INTO Personne(nom, prenom, age)"+
                                        "VALUES('"+N+"','"+P+"','"+A+"')";

st.executeUpdate(requete1);

}catch (SQLException e1) { e1.printStackTrace();}
}
}
```

LES REQUÊTES DE MISES À JOUR

- **Modification** d'un enregistrement :

```
try{
String requete2= "update Personne set age=21 where prenom='mohammed'";
int b =st.executeUpdate (requete2);
System.out.println( " Enregistrement modifiée.");
}catch(Exception ex){
         System.out.println("Erreur: "+ex);}
                    }
```

LES REQUÊTES DE MISES À JOUR

- **Modification** de la base de données :
  - On peut ajouter une nouvelle table Personne à la BD :
  - La table Personne contient 4 attributs et une clé primaire non nulle et auto-incrémentée.

```
try {
  st = con.createStatement();
  String requete3 = "CREATE TABLE Personne(id int not null auto_increment,nom
            VARCHAR (15), prenom varchar (15), age(INT), primary key(id))";

  st.executeUpdate(requete3);

}catch(Exception ex){
                     System.out.println("Erreur: "+ex);
                    }
    }
```

LES REQUÊTES DE MISES À JOUR

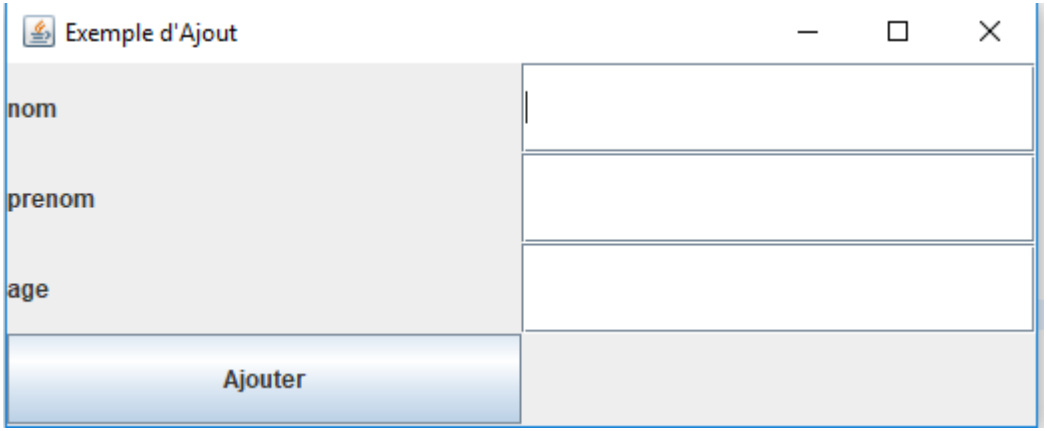
- **Suppression** d'un enregistrement :

```
try{
String requete4 = "delete from Personne where prenom = 'mohammed'";
st.executeUpdate(requete4);
System.out.println("Enregistrement supprimé. ");
}catch(Exception ex){
System.out.println("Erreur: "+ex);}
}
```

Exemple applicatif

LES REQUÊTES DE MISES À JOUR (AJOUT)

- **Ajout** d'un enregistrement :



LES REQUÊTES DE MISES À JOUR (AJOUT)

❑ **Interface Graphique**

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;
import com.mysql.jdbc.Connection;

public class Exemplejout
extends JFrame implements ActionListener
{
JLabel nom,prenom, age;
JTextField nomT, prenomT,ageT;
JButton Ajoute; JPanel j1;
String nom_, prenom_,age_;
Connection con;
public Exemplejout( ) {
setTitle("Exemple d'Ajout");
setSize(530, 220);
setLocationRelativeTo(null);
//Labels
 nom = new  JLabel("nom");
 prenom = new  JLabel("prenom");
 age = new  JLabel("age");
```

```
//TextFields
 nomT= new JTextField(6);
 prenomT =new JTextField(6);
 ageT= new JTextField(6);
//Boutton d'ajout
Ajoute = new JButton("Ajouter");

//Paneau
 j1= new JPanel();
 j1.setLayout(new GridLayout(4, 2));
  j1.add(nom); j1.add(nomT);
 j1.add(prenom); j1.add(prenomT);
 j1.add(age); j1.add(ageT);
 j1.add(Ajoute);
 add(j1);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Ajoute.addActionListener(this);
}
```

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

❑ **Ajout** d'un enregistrement :

```
public void actionPerformed(ActionEvent e) {

//Recupération des données entrées
String nom= nomj.getText();
String prenom= prenomj.getText();
int age=Integer.parseInt(agej.getText());

// établir la Connexion et traiter les données
if(e.getSource()== Ajoute){
     try{
          String url ="jdbc:mysql://localhost/BDTest";
          Class.forName("com.mysql.jdbc.Driver");
          con =DriverManager.getConnection(url,"root","");
          Statement st = con.createStatement();
          System.out.println("connexion  est bien etablie");
          String requete = "INSERT INTO Personne
                            VALUES('"+nom_+"','"+prenom_+"','"+age_+"')";
          int nb=st.executeUpdate(requete);
          System.out.println( "Enregistrement Ajouté.");
     }catch(Exception e1){
     e1.printStackTrace();}
}
```

Aller plus loin

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

❑ **Modification** d'un enregistrement :

```
public void actionPerformed(ActionEvent e) {
String nom= nomj.getText();
String prenom= prenomj.getText();
int age=Integer.parseInt(agej.getText());
System.out.println(""+nom+" "+prenom+" "+age); //tester

if(e.getSource()==modif){

try{//Connexion
     String url ="jdbc:mysql://localhost/BDTest";
     Class.forName("com.mysql.jdbc.Driver");
     con =DriverManager.getConnection(url,"root","");
     Statement st = con.createStatement();
     System.out.println("connexion  est bien etablie");
     String requete= "update Personne set age=21 where prenom='mohammed'";
     st.executeUpdate (requete);
     System.out.println( "Enregistrement modifiée.");
     }catch(Exception e1){
     e1.printStackTrace();}
}
```

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

- **Supprimer** d'un enregistrement :

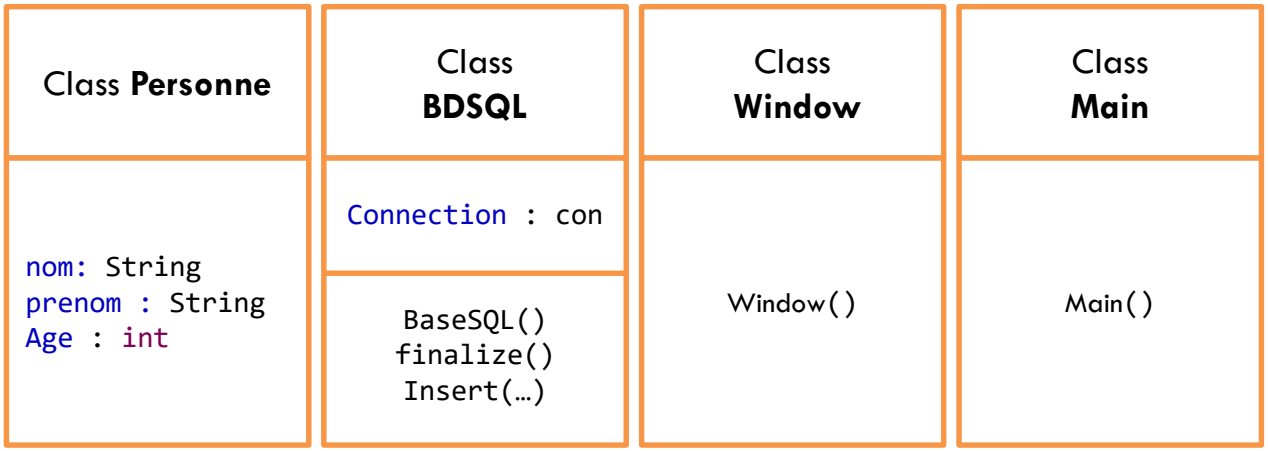
```
public void actionPerformed(ActionEvent e) {
String nom= nomj.getText();

if(e.getSource()==supp){

try{//Connexion
     String url ="jdbc:mysql://localhost/test";
     Class.forName("com.mysql.jdbc.Driver");
     con =DriverManager.getConnection(url,"root","");
     Statement st = con.createStatement();
     System.out.println("connexion  est bien etablie");
     String requete = "delete from Personne where nom =  '"+ nom +"'  ";
     st.executeUpdate(requete);
     System.out.println("Enregistrement supprimé. ");
     }catch(Exception e1){
     e1.printStackTrace();}
}
```

Amélioration

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

- Remarque : On remarque qu'il y a du redondance dans la partie de la connexion pour chaque bouton, pour ce fait, on va créer une classe nommée **BDSQL** qui va retourner un élément de type

- Schéma Global :



LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

- **BASESQL**

```
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class BaseSQL {
private Connection con;
public BaseSQL(){
     try{
          Class.forName("com.mysql.jdbc.Driver");
          String url="jdbc:mysql://localhost/javadb";
          con=DriverManager.getConnection(url,"root","");
          System.out.println("Connexion bien");
         }catch(Exception ex){
                    System.err.println("Erreur:"+ex);
        }
                 }
```

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

- **BASESQL**

```
public void Insert(Personne P) throws SQLException {
String requete1 ="INSERT INTO Personne Value('" + P.nom + "','" +
P.prenom + "'," + P.age + ")";
//System.out.println(requete1);
Statement st = con.createStatement();
st.execute(requete1);
}


@Override
protected void finalize() throws Throwable {
super.finalize();
if(con == null) return;
con.close();
System.out.println("connextion close");

}

}
```

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

- **Personne**

```
public class Personne {
public String nom,prenom;
public int age;
                        }
```

- **Main**

```
public class Main {

public static void main(String[] args) {
new Window();
                                        }
                    }
```

LES REQUÊTES DE MISES À JOUR (AJOUT : 2ÉME FAÇON)

❑ **Window**

```
import java.awt.GridLayout;
import java.awt.event.*;
import java.sql.SQLException;
import javax.swing.*;
public class Window extends JFrame {
private JTextField  j1 = new JTextField ("");
private JTextField  j2 = new JTextField ("");
private JTextField  j3 = new JTextField ("");
private JButton ajouter = new JButton("Ajouter");
JPanel pan = new JPanel();

public Window(){
setSize(300,300);setTitle("Exemple de Mysql");
pan.setLayout(new GridLayout(4,2));
pan.add(new JLabel("Nom:")); pan.add(j1);
pan.add(new JLabel("Prenom:"));pan.add(j2);
pan.add(new JLabel("Age:")); pan.add(j3);
pan.add(ajouter);
```

```
ajouter.addActionListener(new
ActionListener(){
@Override
public void actionPerformed(ActionEvent o){
Personne p = new Personne();
p.nom = j1.getText();
p.prenom = j2.getText();
p.age = Integer.parseInt(j3.getText());

BaseSQL b = new BaseSQL();
try {
b.Insert(p);
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} finally {
b = null;
System.gc();
} });
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
```

Merci pour votre attention