Rappel générale :

Enoncé :

Il vous est demandé de réaliser une base de données permettant de stocker des recettes de cuisine et gérant également les stocks des ingrédients que vous possédez. Chaque recette de cuisine a un nom, une description, la durée de préparation et la durée de cuisson, le nombre de calories par personne, le nombre de parts et le niveau de difficulté : difficile, moyen ou facile. Pour chaque recette vous voulez savoir quels sont les ingrédients nécessaires et la quantité associée à chaque ingrédient. Pour chaque ingrédient vous avez son nom et le nombre de calories pour 100 grammes de cet ingrédient. Chaque ingrédient a un type, par exemple féculent pour l'ingrédient pomme de terre. Un même ingrédient peut avoir plusieurs conditionnements, par exemple, l'ingrédient farine peut-être stocké sous forme d'un paquet de 1 kg ou de 500 g. Ces deux conditionnements seront considérés comme des produits différents. Pour gérer les stocks des ingrédients, le lieu de stockage des produits dans le logement est mémorisé. Chaque rangement est nommé et pour chaque rangement vous savez quels sont les produits qui y sont stockés. Un même ingrédient peut être stocké dans plusieurs rangements.

Travail à Faire :

Faire le diagramme UML de modélisation de données puis la base de données relationnelle et le code java de cette application.

Tour de table :

Enoncé :

Remplir chaque table de l’exercice précédent avec du contenu de votre choix

Et exécuter une requête de recherche ou de filtrage

La connexion à la B.D :

```
import java.sql.DriverManager;
public class Connexion {
com.mysql.jdbc.Connection conn;
{
String url= "jdbc:mysql://localhost:3306/";
String dbName;
dbName = "base";
String driver= "com.mysql.jdbc.Driver";
String userName= "root";
String password="";
try
{
Class.forName(driver).newInstance();
conn= (com.mysql.jdbc.Connection)
DriverManager.getConnection(url+dbName,userName, password);
System.out.println("la connexion a reussie");
}catch(Exception e)
{
System.out.println("la connexion est echoué");
}
}}
```

L'ajout dans une B.D :

```
try {
// TODO add your handling code here:
Connexion c= new Connexion();
java.sql.PreparedStatement statement = c.conn.prepareStatement("INSERT INTO `table`( `champ1`,
`…`) VALUES ('"+valeur1+"','"+ +"')");
statement.executeUpdate();
JOptionPane.showMessageDialog(null,"Ajouté avec
succés","",JOptionPane.INFORMATION_MESSAGE);

} catch (SQLException ex) {
JOptionPane.showMessageDialog(null, ex); }
```

La modification des données :

```
try {
// TODO add your handling code here:
Connexion c= new Connexion();
String sql="UPDATE `table` SET `ch1`='"+v1+"' ,`chn`='"+vn+"' WHERE
`id`='"+v_id+"';";
java.sql.PreparedStatement statement = c.conn.prepareStatement(sql);
statement.executeUpdate();
;
JOptionPane.showMessageDialog(null,"Modifié avec
succés","",JOptionPane.INFORMATION_MESSAGE);
} catch (SQLException ex) {
JOptionPane.showMessageDialog(null, ex); }
```

La suppression des données :

```
try {
// TODO add your handling code here:
Connexion c= new Connexion();
String sql = "DELETE FROM `table` WHERE `id`=?";
java.sql.PreparedStatement statement = c.conn.prepareStatement(sql);
statement.setInt(1, valeur_de_id);
statement.execute();
JOptionPane.showMessageDialog(null,"Supprimé avec
succés","",JOptionPane.INFORMATION_MESSAGE);
aff();
} catch (SQLException ex) {
JOptionPane.showMessageDialog(null, ex); }
```

Affichage des données : (affichage des formations)

```
private static void aff() throws SQLException
	{
	Connexion c=new Connexion();
	PreparedStatement pst;
	pst = (PreparedStatement) c.conn.prepareStatement("SELECT * FROM
formation");
	pst.executeQuery();
	ResultSet rs = (ResultSet) pst.executeQuery();
	while(rs.next())
	{
		System.out.println(rs.getString("titre"));
	}
	}
```