Lab Test 1: Coffee Café Sales

CoffeeApp is a standalone application which allows a coffee café owner to update his monthly sale of a selected coffee brand. Input of i) a brand name, ii) supplier id, iii) price per kg and iv) no of coffee sales (by kg – accepted input whole number eg: 40) are received via dialog boxes. The app calculates the total sales for the brand and update these data in the database by creating a record for the specific brand in table coffee.

Requirement:

1. Installation **https://netbeans.org/downloads/index.html**
   - Install NetBeans IDE Bundles JAVA EE (inclusive of web server)
2. You will need derbyclient.jar which contains driver to connect to Apache Java Derby. It comes with Java JDK and is usually found in the following directory (note under database libraray directory)

   C:\Program Files\Java\jdk1.8.0_45\db\lib (for example)

This test is divided into 2 parts, part 1 is configuring a database and part 2 is developing the CoffeeApp using the NetBeans IDE.

Part 1: Configuring database

Step 1: Creating database

In this lab test you will need to perform a derby database configuration by creating a database and a table. At the end of Part 1 you should get your database configured as depicted by the following figure 1:

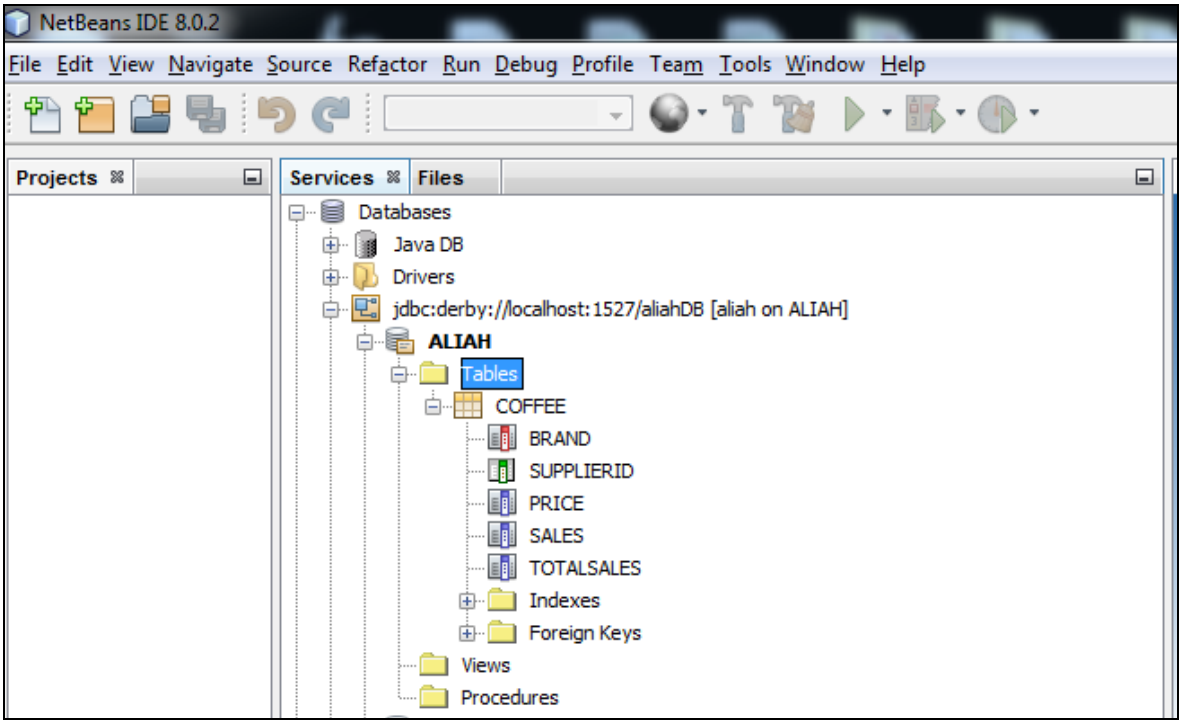
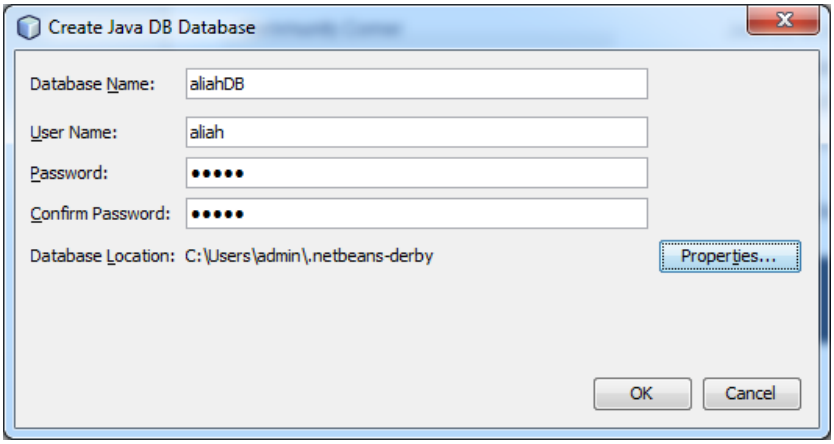
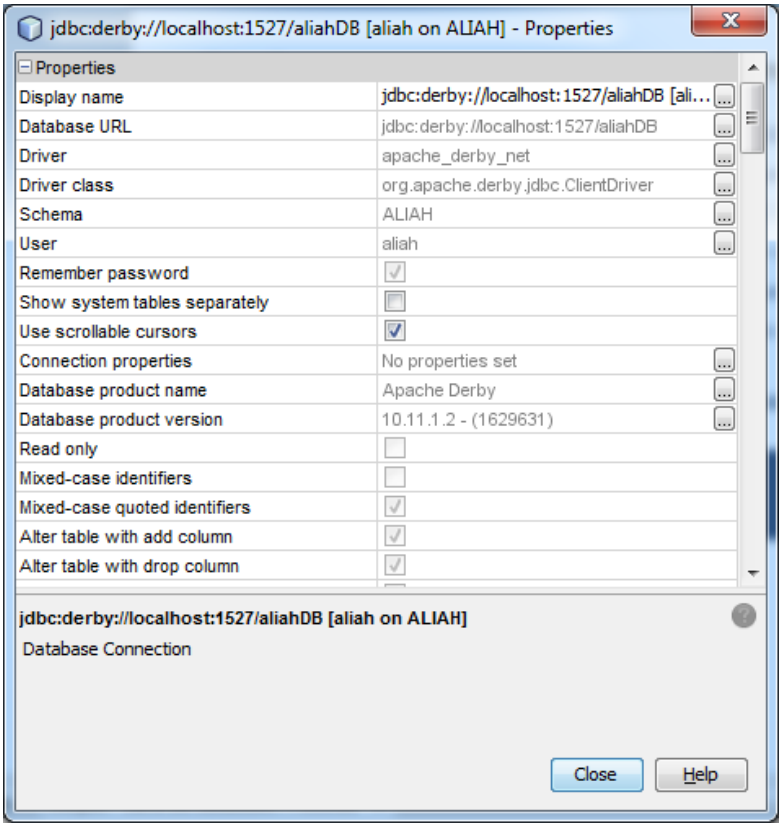


Figure 1

- Select tab **Services**>rightclick **Java DB**> click **Start Server** (to start the Apache Derby Network Server)
- rightclick **Java DB**> click **Create Database** (to create a new database – use your name as the **database name**, **username** and **password**) as follows:



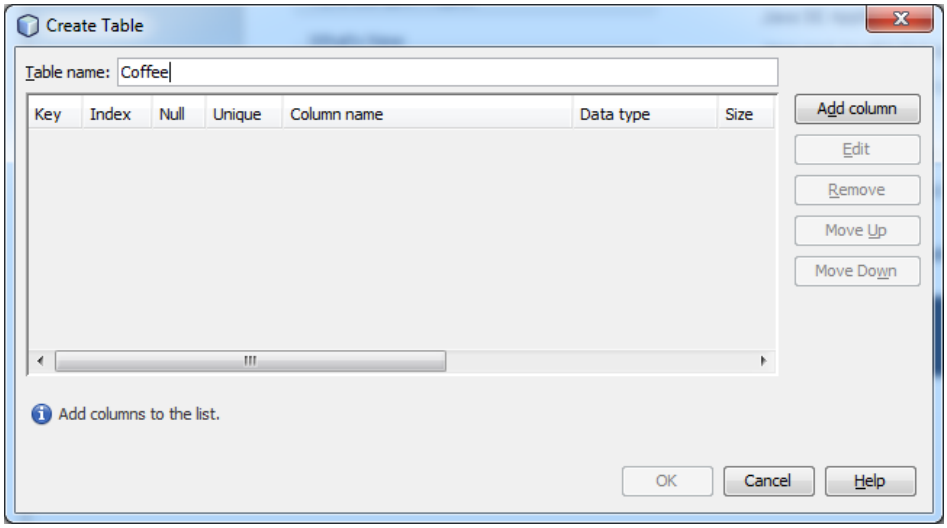
- Right click on the connection of your newly created database and click **connect** (to start using it)
- To check the properties of your database, right click on your database connection and click **properties**, the properties window will open as follows:



- Click on Database URL and Driver Class to get the info which you will need in developing the CoffeeApp.

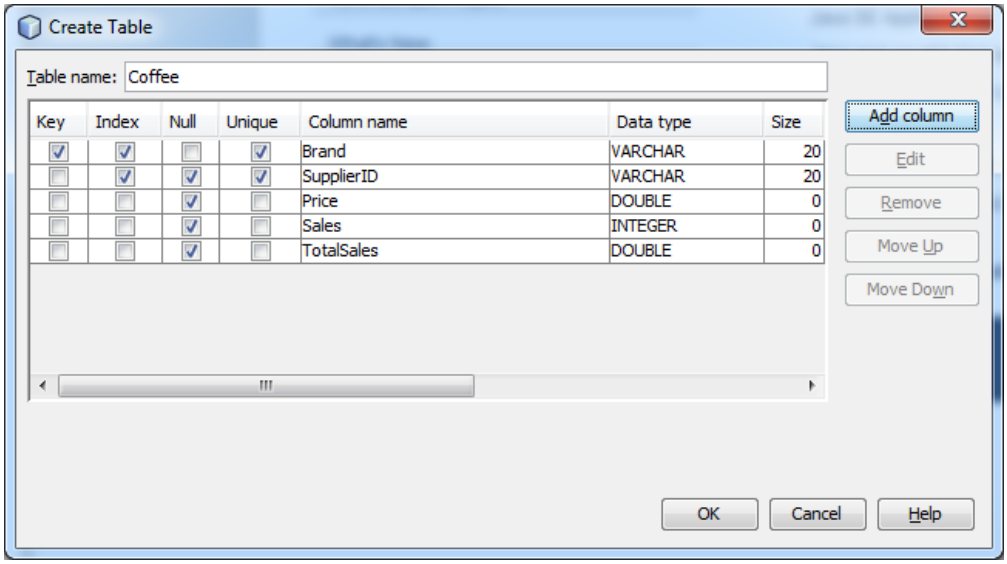
Step 2: Creating the Coffee table

- Explore your database and you will see 3 tabs (Tables, Views and Procedures)
- Right click Tables>click **Create Table**



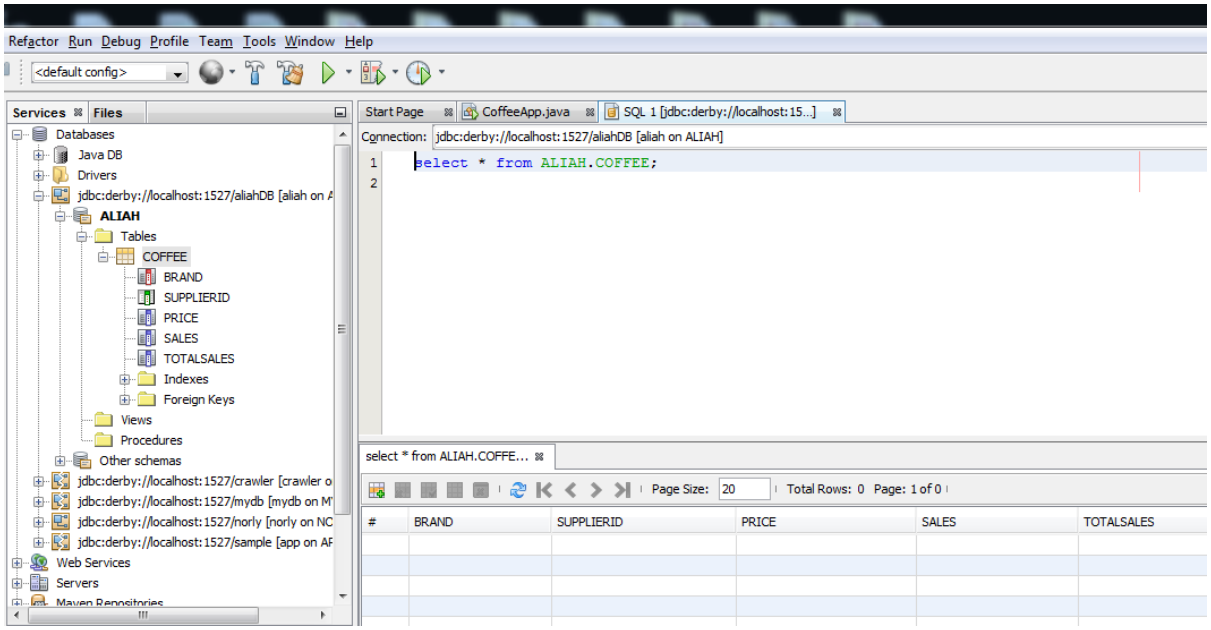
- Add column as follows:
  - Brand – db type: varchar 20, primary key + unique
  - SupplierID – db type: varchar 20, unique
  - Price – db type: double
  - Sales – db type: int
  - TotalSales – db type: double

  Click OK as follows:



  A table named Coffee will be created and configured as required (as in **Figure 2**)

- Right click **Coffee** table>click **View Data**. SQL command **select * from ALIAH.COFFEE;** is executed which shows the content of the **Coffee** table.

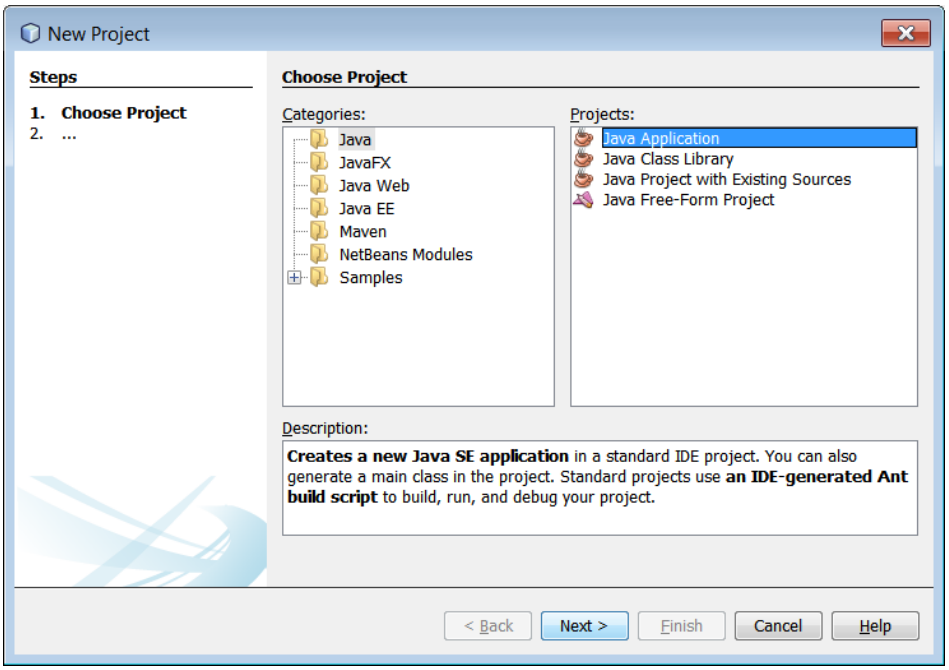


Part 2: Creating CoffeeApp

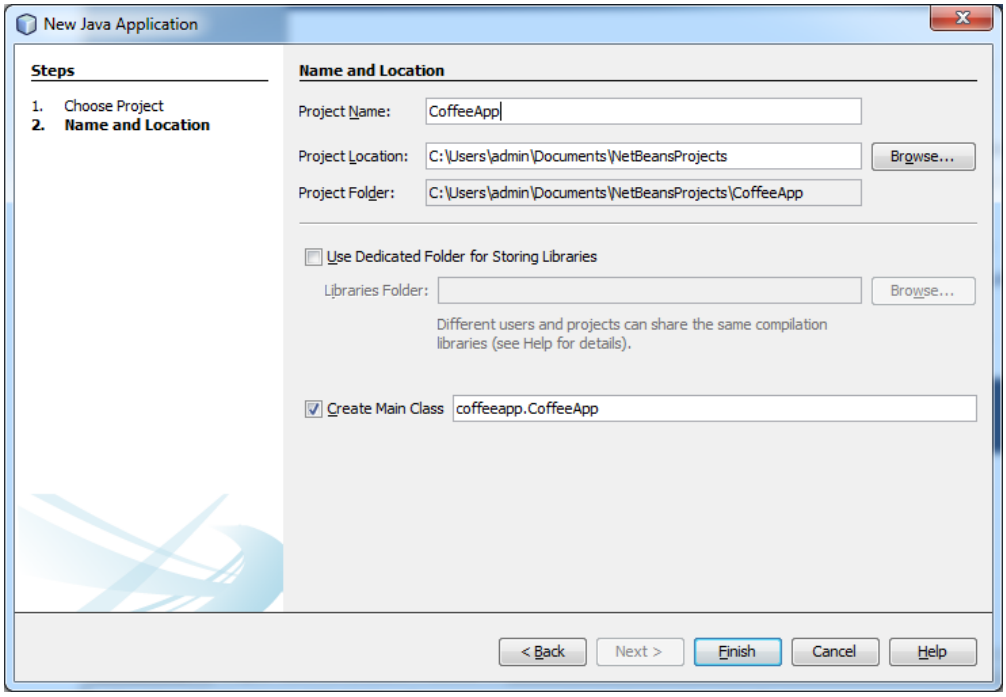
Step 1: Creating a new java project

In this lab test you will need to modify the code provided to create **CoffeeApp** which will connect to the database created in part 1.

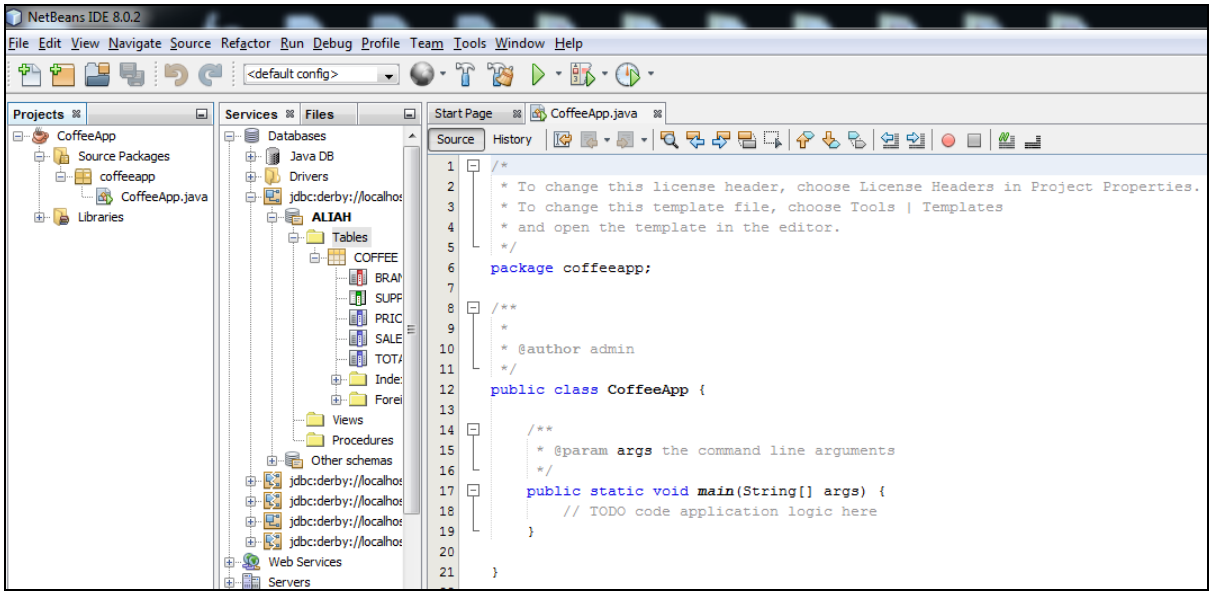
- Click File->New Project, select Categories **Java**>Projects **Java Application**, and click Next.



- Enter project name **CoffeeApp**, check the checkbox to Create

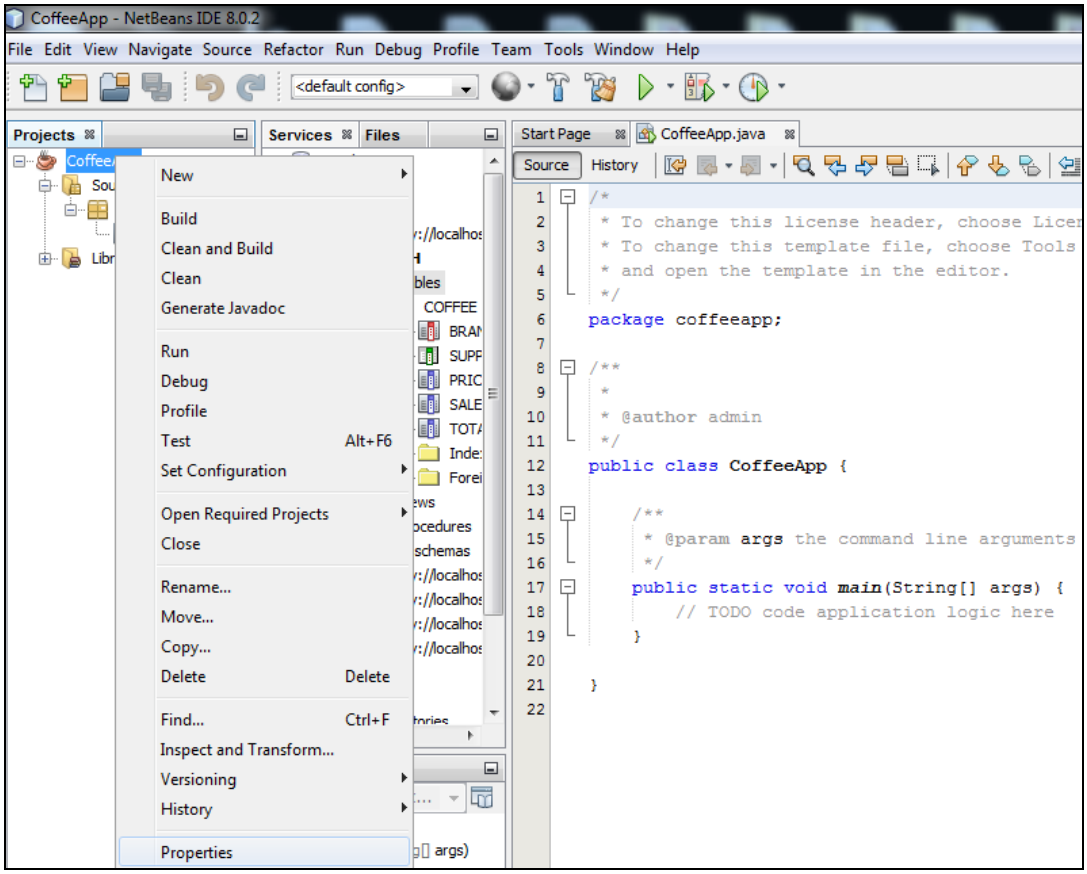


- Click tab Projects, and explore CoffeeApp project, you will find CoffeeApp.java program in **coffeeapp** package.

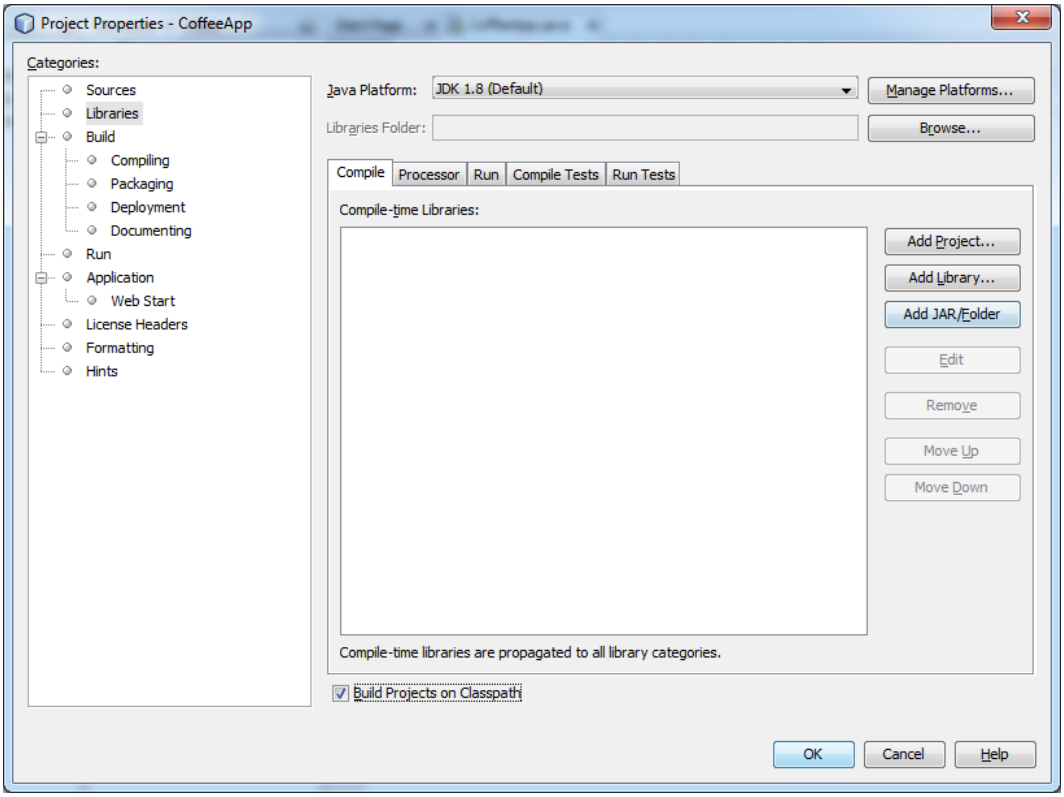


Step 2: Adding derbyclient.jar to project library

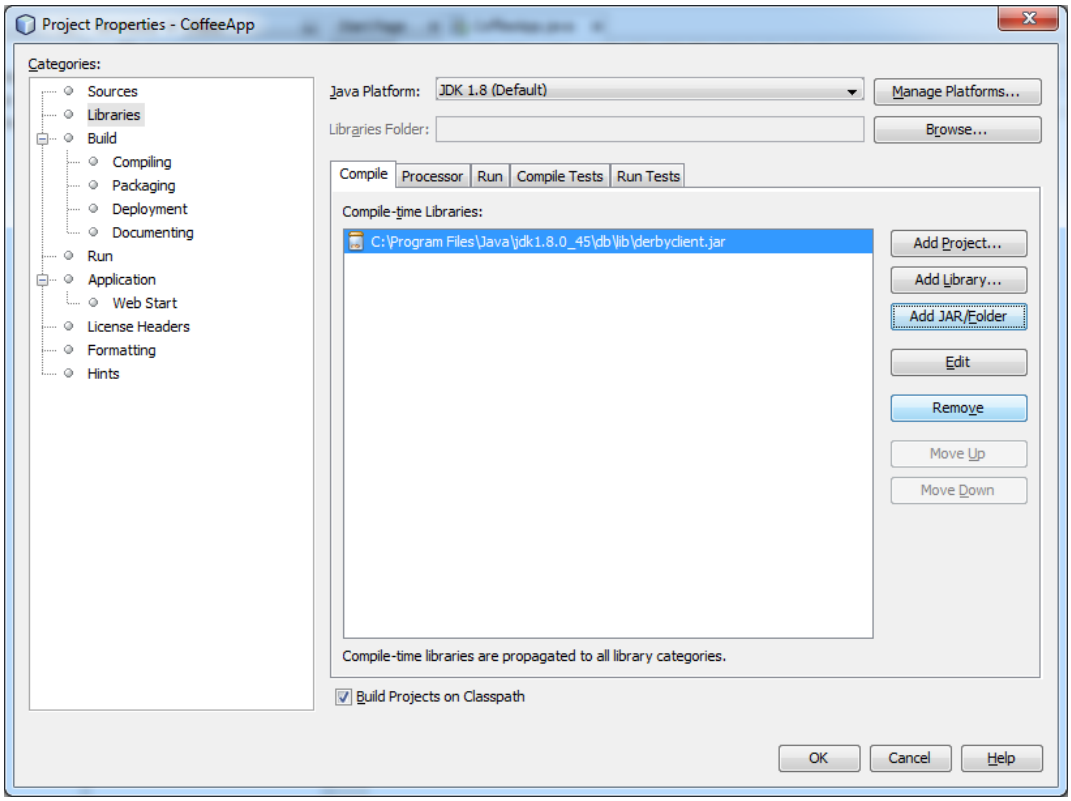
- This step is necessary to enable CoffeeApp to find the database driver required.
- In Projects tab, right click project name (**CoffeeApp**) > select **Properties**.



- Project Properties window appears. The Categories on left side, select **Libraries**. And on right side in Compile tab, click **Add JAR/Folder**.



- Browse to **C:\Program Files\Java\jdk1.8.0_45\db\lib** directory and select **derbyclient.jar**. You'll see the derbyclient.jar file is added to the project. Click OK to finish.



Step 3: Modify program provided to complete CoffeeApp

- Copy the content of the program provided into CoffeeApp.java
- Add import statement `import java.util.Properties;`
- Modify the following `dbUrl` to point to your database url (check Properties)

```
String dbUrl = "jdbc:mysql://localhost:3306/aliahDB";
```

- Modify the following `driver` to reflect the driver class of your database (check Properties)

```
String driver = "org.mysql.jdbc.ClientDriver";
```

- Modify the following queries to reflect your table name:

```
String query1 = "INSERT INTO ALIAH.COFFEE VALUES(?, ?, ?, ?, ?)";
String query2 = "SELECT * FROM ALIAH.COFFEE";
```

- Add the following codes in the main method to use `java.util.Properties` class to create `userinfo` object to configure `username` and `password`.

```
Properties userInfo = new Properties();
userInfo.put("user", "aliah");
userInfo.put("password", "aliah");
```

- Modify the following codes to only receive 4 input (brand, supplier id, price and sales)

```
String name = JOptionPane.showInputDialog("Coffee Brand");
String supplier = JOptionPane.showInputDialog("SupplierId ");
String priceKg = JOptionPane.showInputDialog("Price/kg(RM) ");
String salesKg = JOptionPane.showInputDialog("Sales (kg) ");
String totalSales = JOptionPane.showInputDialog("TotalSales(RM)");
```

- Add codes to calculate the total sales (price x sales). Note that you have to change String to Double and Integer data type to perform arithmetic operation. (Hint: use `parseDouble()` and `parseInt()` methods)

- Modify the following codes for `DriverManager` to create `theConnection` object. Pass `dbUrl` and Properties object (`userInfo`) instead of passing user name and password as parameter to method `getConnection()`.

  ```
  theConnection = DriverManager.getConnection(dbUrl,"root", "");
  ```

- Modify the following codes to set the appropriate values into the query

```
//Inserting data into table in database
st1 = theConnection.prepareStatement(query1); //step3
st1.setString(1, name);
st1.setString(2, supplier);
st1.setString(3, priceKg);
st1.setString(4, salesKg);
st1.setString(5, totalSales);
st1.executeUpdate();
```

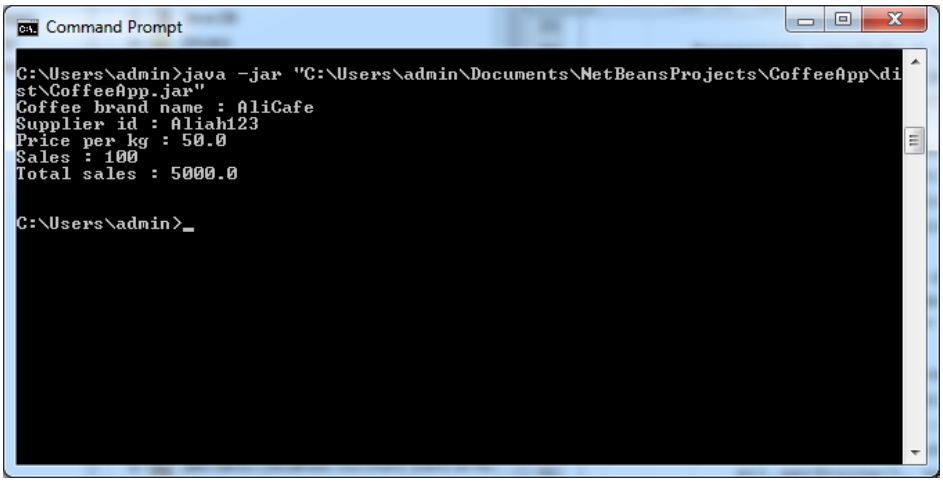
- Modify the following codes to output the appropriate values based on data types

```
while (result.next()){
   System.out.println("Coffee brand name : "+result.getString(1));
   System.out.println("Supplier id : "+result.getString(2));
   System.out.println("Price per kg : "+result.getString(3));
   System.out.println("Sales : "+result.getString(4));
   System.out.println("Total sales : "+result.getString(5));
   System.out.println();
}
```

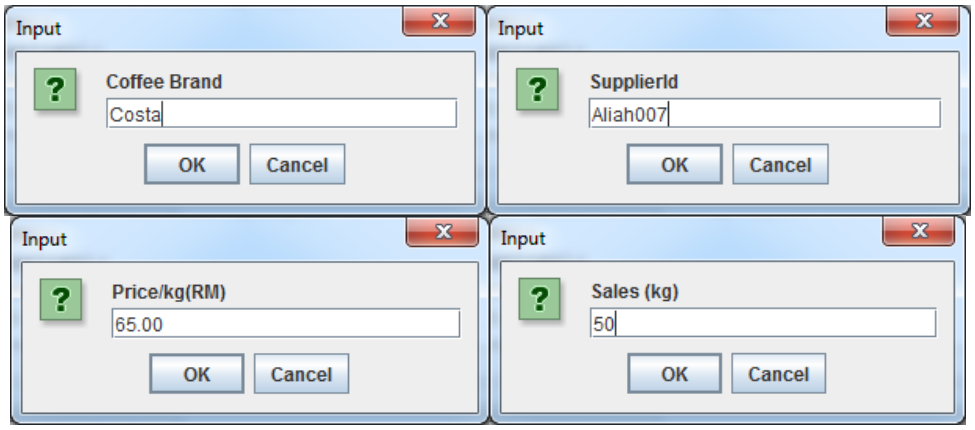
Step 4: Compile and run

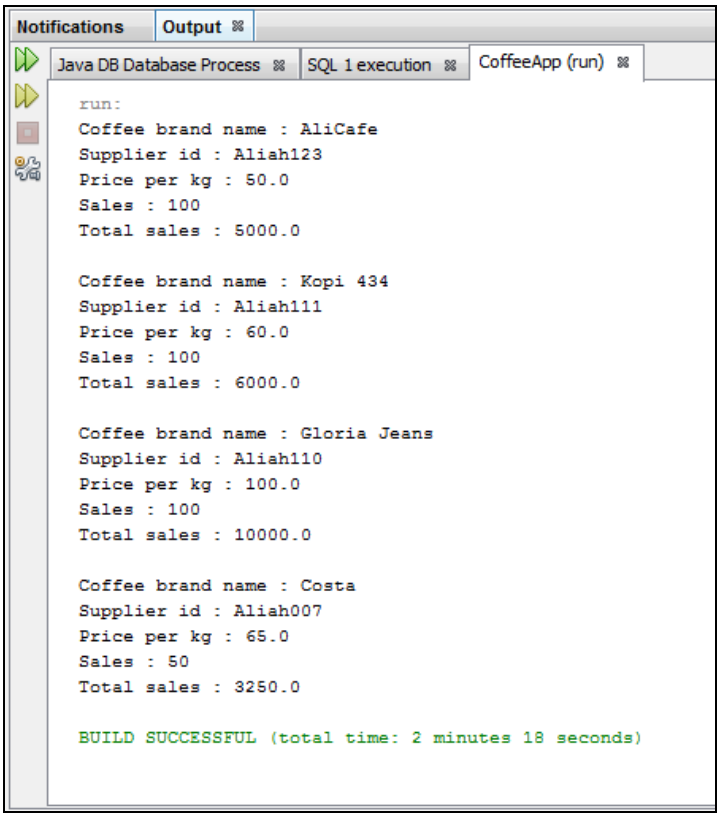
- Right click project name (CoffeeApp)> click Clean and Build.
- To run this application from the command line without Ant, try:

  java -jar "C:\Users\admin\Documents\NetBeansProjects\CoffeeApp\dist\CoffeeApp.jar"



- Or run in NetBeans by right click> Run File or Run Button:





- Right click table **Coffee**> **View Data** to see records have been added into your table.