

JDBC

Java Data Base Connectivity.

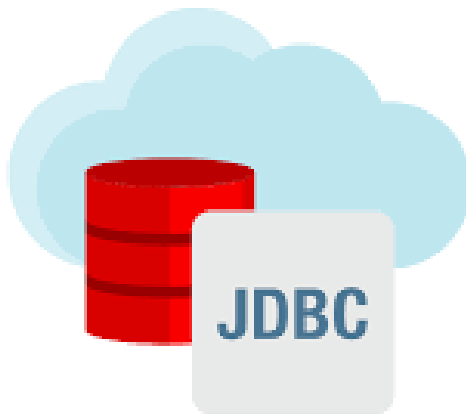
BY:

- Omar Mohamed Emam 44
- Islam Yousry 14
- Islam Mohamed 12
- Bahaa Khalaf 21



Report Content:

1. Program Specifications
2. User Guide
3. UML Diagrams
4. Program Description
5. Design Patterns
6. Sample Runs

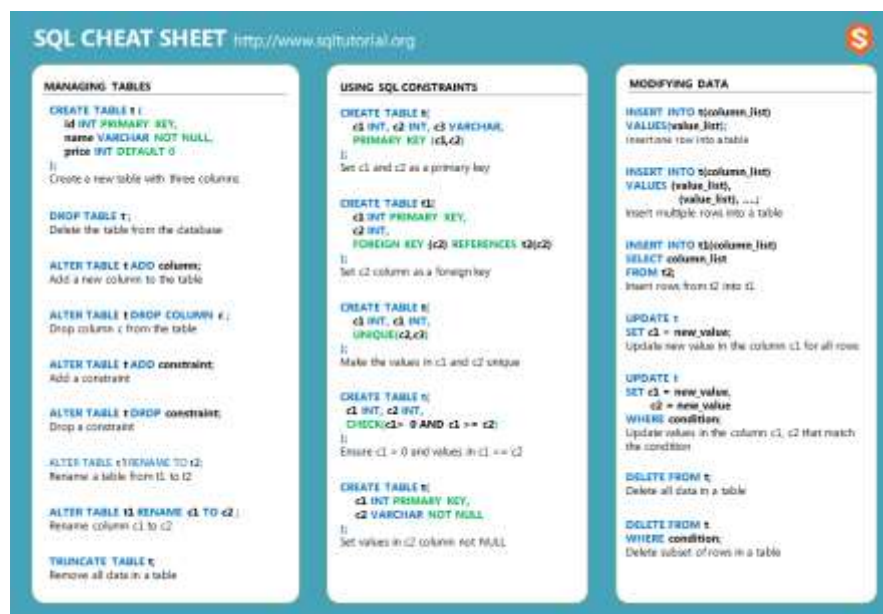


Program Specifications:

- Program creates the driver , connection , statement which are ready to accept your sql query , check it and then transfer it to DBMS to execute it and send the result back (if there is a result) or just edit or create your databases and your tables
- Program deals with URL as database path in your local machine, so there is no existence for servers and its functions
- Program will keep running even if you entered in wrong syntax, you will be able to retry till you want to stop
- Program runs through self-runnable Cmd app

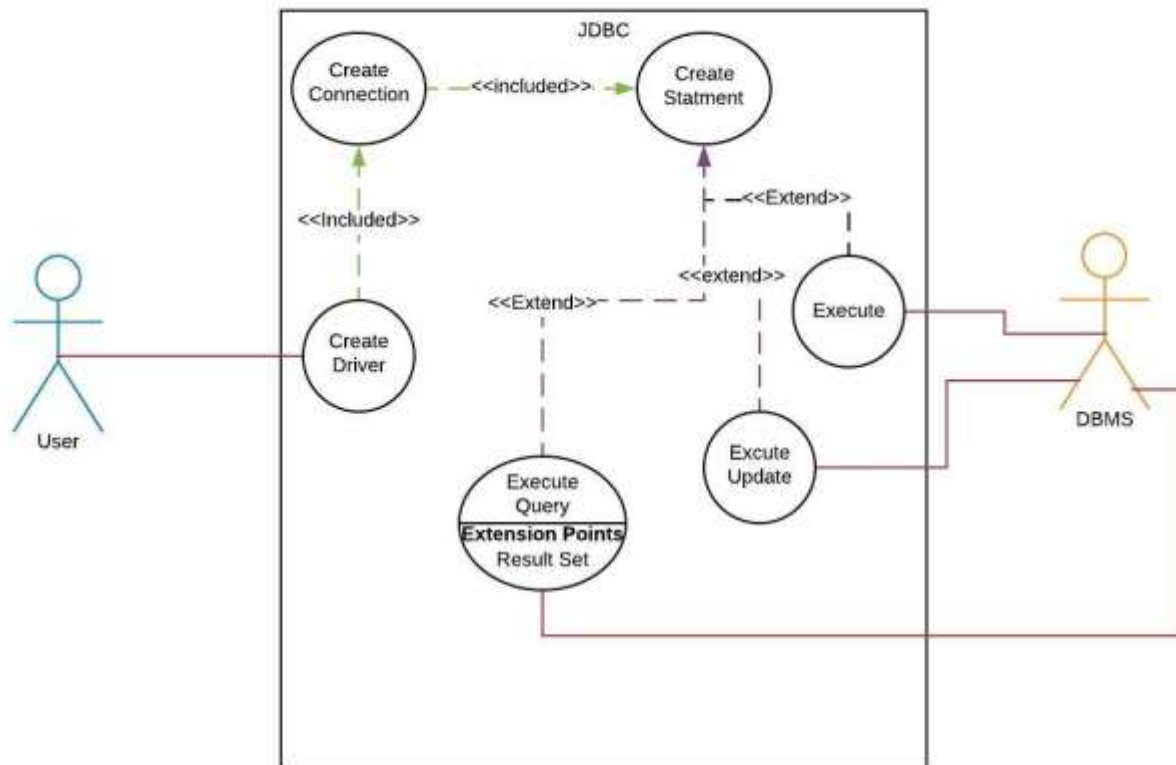
User Guide:

Run “excucation.bat”, wait creation of connection then enter your SQL query in right syntax and then you can do more queries or exit by typing “end”.



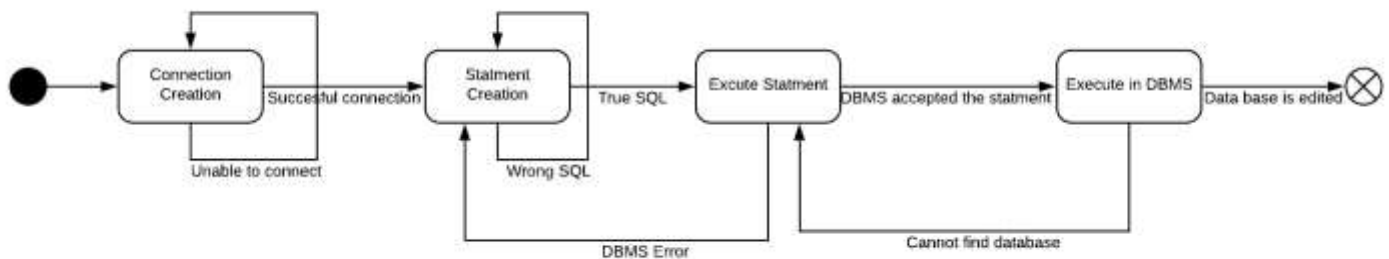
UML Diagrams:

- Use Case Diagram :



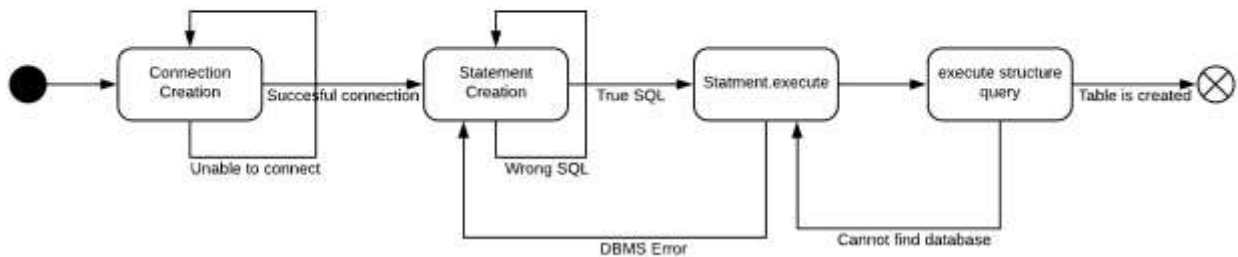
- State Diagram For 3 Scenarios :

- ** General Scenario:



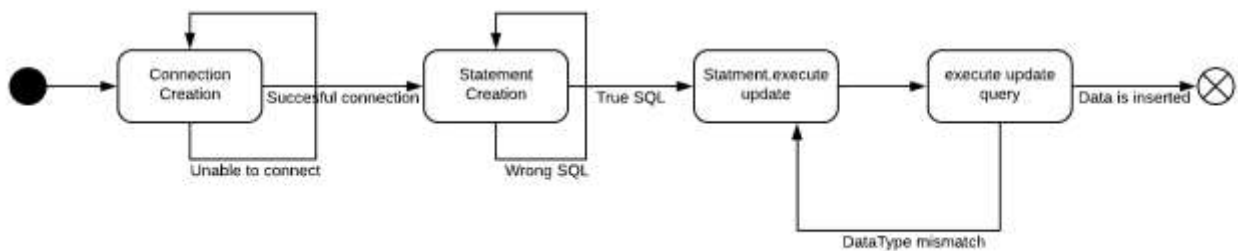
**** Scenario #1 (create table):**

CREATE TABLE



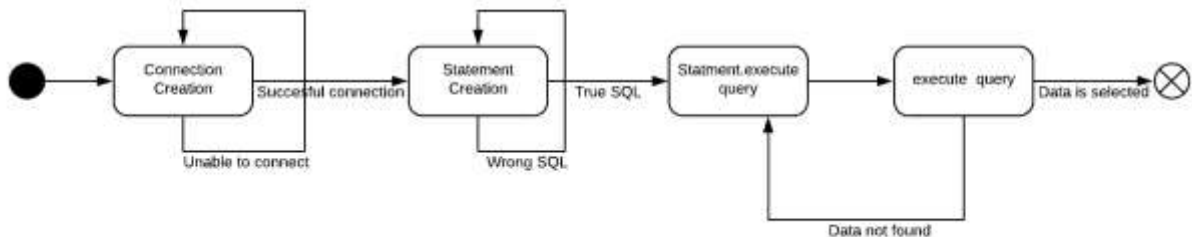
**** Scenario #2 (insert into table):**

INSERT INTO TABLE

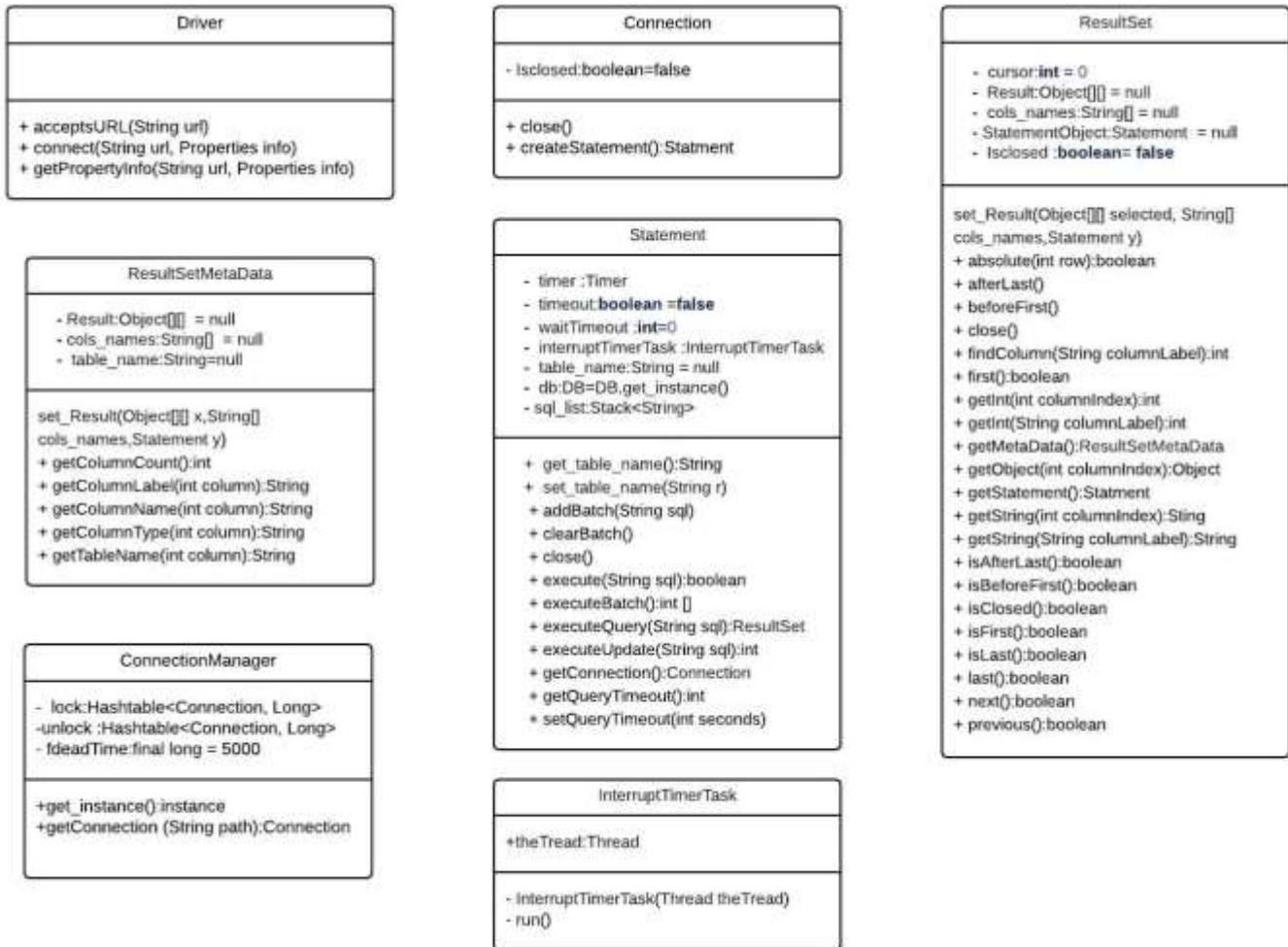


**** Scenario #3 (select from table):**

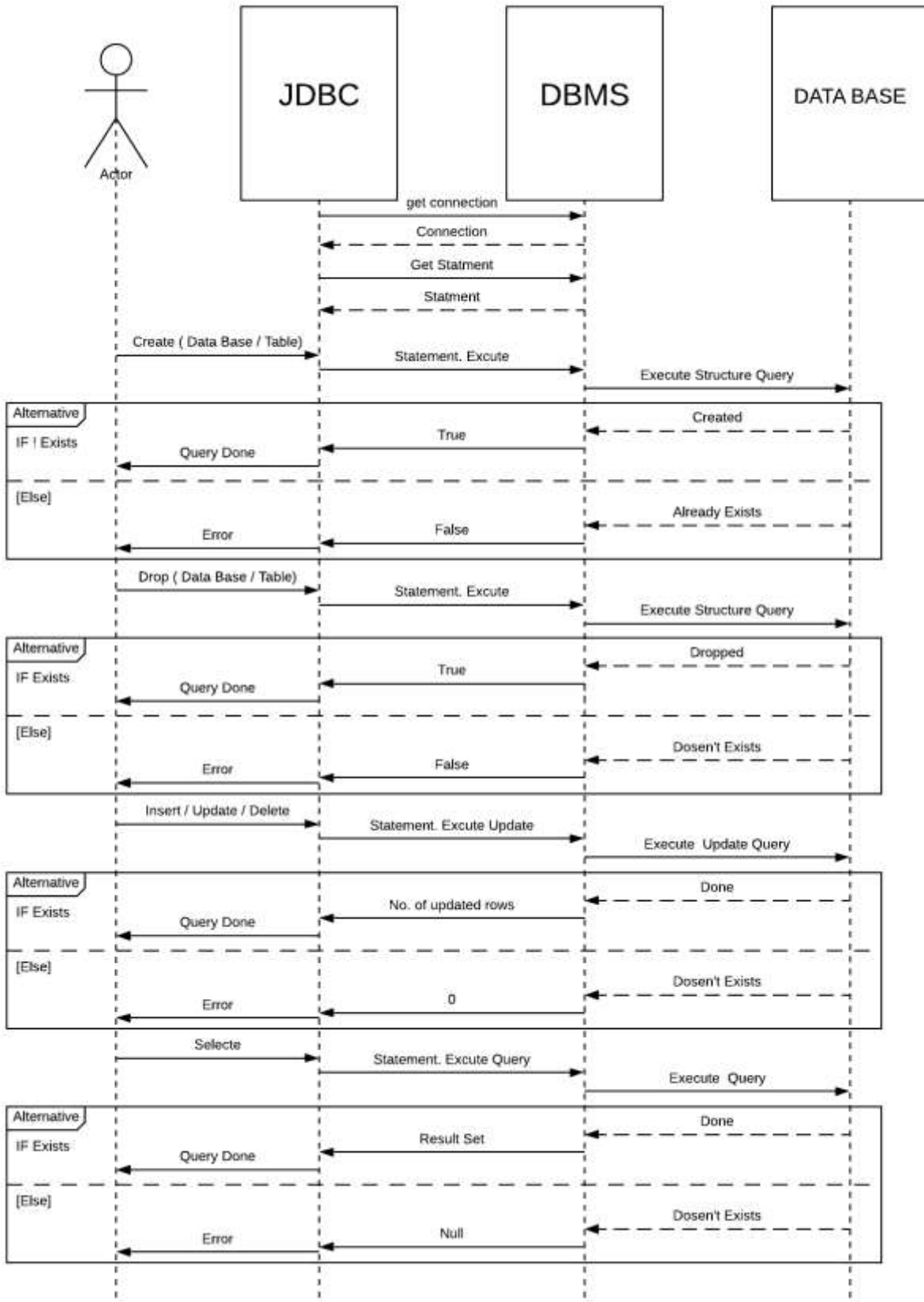
SELECT FROM TABLE



• Class Diagram :



• Sequence Diagram :



Program Description:

We implemented some function in JDBC interfaces as following:

In `java.sql.Driver`:

- **Accept URL**: takes the url which must be path of data base and check if it's correct or not
- **Connect**: Create a new connection between the driver and the database
- **Get property Info**: Gets information about the possible properties for this driver.

In `java.sql.Connection`:

- **Crete Statement**: Creates a Statement object that will generate ResultSet objects with the given type and concurrency.
- **Close**: Releases this Connection object's database and JDBC resources immediately instead of waiting for them to be automatically released.

In `java.sql.Statement`:

- **Add Batch**: check if sql query is right in syntax or not and add it to stack
- **Clear Batch**: Remove all queries from the stack
- **Execute**: Send Sql query to execute structure query function in DBMS
- **Execute Update**: Send Sql query to execute update query function in DBMS
- **Execute Query**: Send Sql query to execute query function in DBMS
- **Set timeout**: makes sure that execute function won't pass the required run time
- **Get timeout**: get execution time for execute functions

In `java.sql.ResultSet`:

- **Absolute** : Moves the cursor to the given row number in this `ResultSet` object
- **After Last**: Moves the cursor to the end of this `ResultSet` object, just after the last row.
- **Before First**: Moves the cursor to the front of this `ResultSet` object, just before the first row.
- **Close**: Releases this `ResultSet` object's database and JDBC resources immediately instead of waiting for this to happen when it is automatically closed.
- **Find Column**: Maps the given `ResultSet` column label to its `ResultSet` column index.
- **First**: Moves the cursor to the first row in this `ResultSet` object.
- **Get Int**: Retrieves the value of the designated column in the current row of this `ResultSet` object as an `Int` in the Java programming language.
- **Get Meta Data**: Retrieves the number, types and properties of this `ResultSet` object's columns.
- **Get Object**: Gets the value of the designated column in the current row of this `ResultSet` object as an `Object` in the Java programming language
- **Get Statement**: Retrieves the `Statement` object that produced this `ResultSet` object.
- **Get String(Int column Index)** : Retrieves the value of the designated column in the current row of this `ResultSet` object as a `String` in the Java programming language.
- **Get String (String column Label)**: Retrieves the value of the designated column in the current row of this `ResultSet` object as a `String` in the Java programming language.
- **Is After Last**: Retrieves whether the cursor is after the last row in this `ResultSet` object.
- **Is Before First**: Retrieves whether the cursor is before the first row in this `ResultSet` object.
- **Is Closed**: Retrieves whether this `ResultSet` object has been closed.
- **Is First**: Retrieves whether the cursor is on the first row of this `ResultSet` object.
- **Is Last**: Retrieves whether the cursor is on the last row of this `ResultSet` object.
- **Last**: Moves the cursor to the last row in this `ResultSet` object.
- **Next**: Moves the cursor forward one row from its current position.
- **Previous**: Moves the cursor to the previous row in this `ResultSet` object.

Design Patterns:

Object Pool design pattern:

Object pool pattern is a software creational design pattern which is used in situations where the cost of initializing a class instance is very high.

Basically, an Object pool is a container which contains some amount of objects. So, when an object is taken from the pool, it is not available in the pool until it is put back.

Singleton design pattern:

The singleton pattern is a design pattern that restricts the instantiation of a class to one object.

Sample Runs:

```
Enter Your Query TO execute it or write (end) to exit :
create database test
DataBase>> test << is Created and Saved
Execution time in ms = 127
Enter Your Query TO execute it or write (end) to exit :
create table name (co1 int , co2 varchar)
Table>> name << is Created
Execution time in ms = 67
Enter Your Query TO execute it or write (end) to exit :
insert into name (co1) values (1000)
You Inserted Into >> name
1
Execution time in ms = 100
Enter Your Query TO execute it or write (end) to exit :
select * from name
selected values:
=====
1000    null
=====
Execution time in ms = 13
Enter Your Query TO execute it or write (end) to exit :
end
Ending Program...
Closing Statement.... Statement Closed
Closing Connection..... Connection closed
..JDBC is Closed
***** see you soon *****

D:\JDBC>pause
Press any key to continue . . .
```

```

D:\JDBC>java -jar JDBC.jar
***** Welcome To JDBS *****
Creating Driver..... Driver Created
Creating Connection..... Connection Created
Creating Statement..... Statement Created
Enter Your Query TO execute it or write (end) to exit :
select * from name
selected values:
=====
1000    null
=====
Execution time in ms = 139
Enter Your Query TO execute it or write (end) to exit :
insert into name (co1 ,co2 ) values (10000 , 'oop')
You Inserted Into >> name
Execution time in ms = 114
Enter Your Query TO execute it or write (end) to exit :
select * from name
selected values:
=====
1000    null
=====
10000   oop
=====
Execution time in ms = 14
Enter Your Query TO execute it or write (end) to exit :
end
Ending Program...
Closing Statement.... Statement Closed
Closing Connection..... Connection closed
..JDBC is Closed
***** see you soon *****
D:\JDBC>pause

```

```

D:\JDBC>java -jar JDBC.jar
***** Welcome To JDBS *****
Creating Driver..... Driver Created
Creating Connection..... Connection Created
Creating Statement..... Statement Created
Enter Your Query TO execute it or write (end) to exit :
drop table name
Table> name << is Dropped
Execution time in ms = 2
Enter Your Query TO execute it or write (end) to exit :
drop database test
DataBase>> test << is Dropped and Deleted
Execution time in ms = 3
Enter Your Query TO execute it or write (end) to exit :
end
Ending Program...
Closing Statement.... Statement Closed
Closing Connection..... Connection closed
..JDBC is Closed
***** see you soon *****

D:\JDBC>pause
Press any key to continue . . .

```