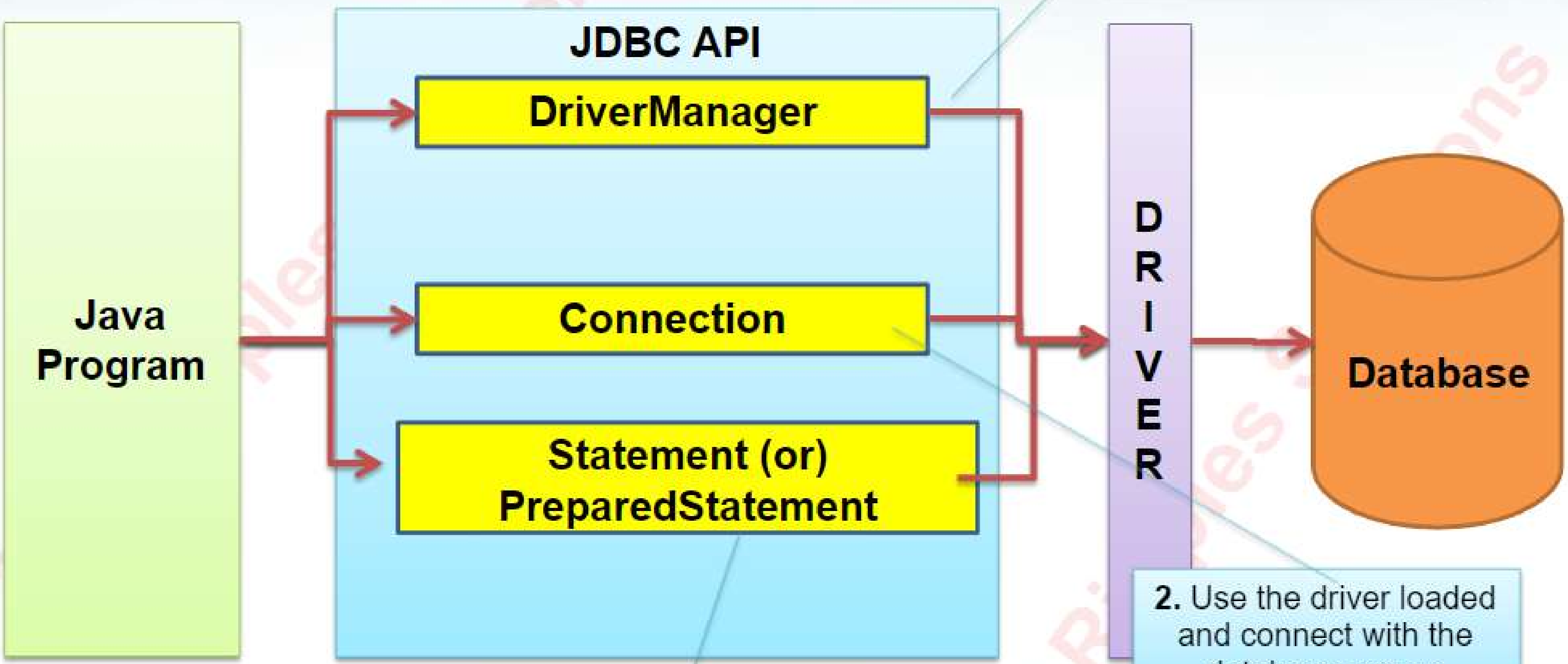
Learning Goals

Click to Continue

This session will help you to understand about,

- Executing a SQL query using JDBC drivers.
- Difference between Statement and Prepared Statement
- Execute queries using statements

How JDBC connect with database? Click to Continue



1. Driver manager loads the driver

2. Use the driver loaded and connect with the database server.

3. Use the statement API to execute the queries

Steps involved In executing query..

Click to Continue

Following are the steps involved in executing a query using JDBC,

1. Import relevant JDBC API's in your java program.
2. Use ***DriverManager*** to load and register the driver class.
3. Use ***Connection*** object to connect to database.
4. Use ***Statement*** (or) ***PreparedStatement*** to create a query.
5. Execute the statement (***or***) prepared statement.
6. Process the ***ResultSet*** *object which has the result of the query executed.*
7. Close the ResultSet and Statement.
8. Close the Connection.

Let us look at the steps

Click to Continue

1.Import the relevant JDBC API's.

Illustration:

```
import java.sql.Driver;
import java.sql.DriverManager;
```

2.The JDBC driver of the appropriate database needs to be placed in the class path and loaded in the java program as below.

Illustration:

```
Driver driver = new com.mysql.jdbc.Driver ();
                    (or )
Class.forName("com.mysql.jdbc.Driver ");
```

The drivers will be provided by the appropriate database vendor

Register & Connect

Click to Continue

3. Loaded driver should be registered with the ***DriverManager***.

Illustration:

```
DriverManager.registerDriver(driver);
// Here driver is the driver object loaded
```

4.Connect with the database server using the driver manager by passing the URL and the credentials.

Illustration:

```
Connection  connection=DriverManager.getConnection(url,username,password);
```

Where,

url represents the URL of the database to be logged on.

userName, represents the user name of a database user with which we can logon.

password, represents the password of the credential.

Connection URL

Click to Continue

URL Illustration:

"jdbc:mysql://[host]:[port]/dbname";

Where,

jdbc:mysql : Represents the driver name

host - The host name or the IP address of the machine where database server is running.

port - The port number where database is listening for connection, default is 1521.

dbname –name of the database instance.

Illustration: String url="**jdbc:mysql://localhost:3306/ripples**"
Where, **ripples** is the database name.

Creating the Statement

Click to Continue

5. Creating the statement: A ***statement*** object is created for executing the query.

Syntax:

```
        Statement stmt = conn.createStatement();
```

Where,

conn - is the connection object created.

6. **Executing the query:** The query is executed using the statement object created. This returns a JDBC ***ResultSet*** interface.

Syntax: For executing a select query

```
String query="select student_name, id from Students";
ResultSet  rs= stmt.executeQuery (query);
```

Syntax: Executing a DML query use ***executeUpdate().***

```
            String query="Insert (or) update query goes in here"
            int  rowCount = stmt.executeUpdate (query);
```

Where**, rowCount** is the number of records impacted by the insert or update query

7: Process data from ResultSet

Click to Continue

What is a ***ResultSet***?

The result set represents a row in a table, a logical grouping to hold the results of a query and access it in a java program.

7: Processing data from ResultSet: Iterating through the result set is done using the ***next()*** API of the **ResultSet** object.

```
while (rs.next())
{
    // Each row is iterated and data accessed.
}
```

Example: The data is retrieved using the ***getter*** methods of the JDBC ***ResultSet*** interface.

```
while (rs.next()) {
    String studentName = rs.getString(1);
    long id = rs.getLong(2)
    }
```

Since student name is string we are using **getString()** for numeric fields use the appropriate getters.

Here the parameters **1, 2** denote the position of the columns in the SQL query.
Select studentname , id from student;

7: Process data from ResultSet

Click to Continue

Instead of passing the column index you can pass the column name as parameter to the ***get*** methods of the JDBC ***ResultSet*** interface

```
while (rs.next()) {
        String studentName = rs.getString(“studentname”);
        long salary = rs.getLong(“id”)
        }
```

Here the parameters of the method are the column names mentioned in the SQL.
Select studentname , id from student;

Last Step: Close the result set and connection in the finally block of the code.

Syntax:

```
rs.close();  // Result set closed

stmt.close(); // Statement closed
```

Syntax:

```
conn.close();

// Connection is closed
```

- Always close connections after executing queries.
- Failure in closing connections will result in database connection leaks which will crash the application.
- Always close connections in finally blocks.

Accessing database in nutshell

Click to Continue

1: Import JDBC relevant API's

```
import java.sql.Driver;
import java.sql.DriverManager;
```

2: Loading the Driver

```
Driver driver=new
com.mysql.jdbc.driver.Driver();
```

3: Register the Driver

```
DriverManager.registerDriver(driver);
```

4: Establishing Connection

```
Connection
connection=DriverManager.getConnection
(url,username,passwd);
```

5: Creating A Statement

```
Statement
statement=connection.createStatement();
```

6: Querying the Database

```
String query=“select  name,age from
student”;
ResultSet rs = stmt.executeQuery(query);
```

7: Processing the Results of the Query

```
while (rs.next()) {
int name= rs.getInt(1);
long sal=rs.getLong(2)
}
```

8: Closing the ResultSet and Statement

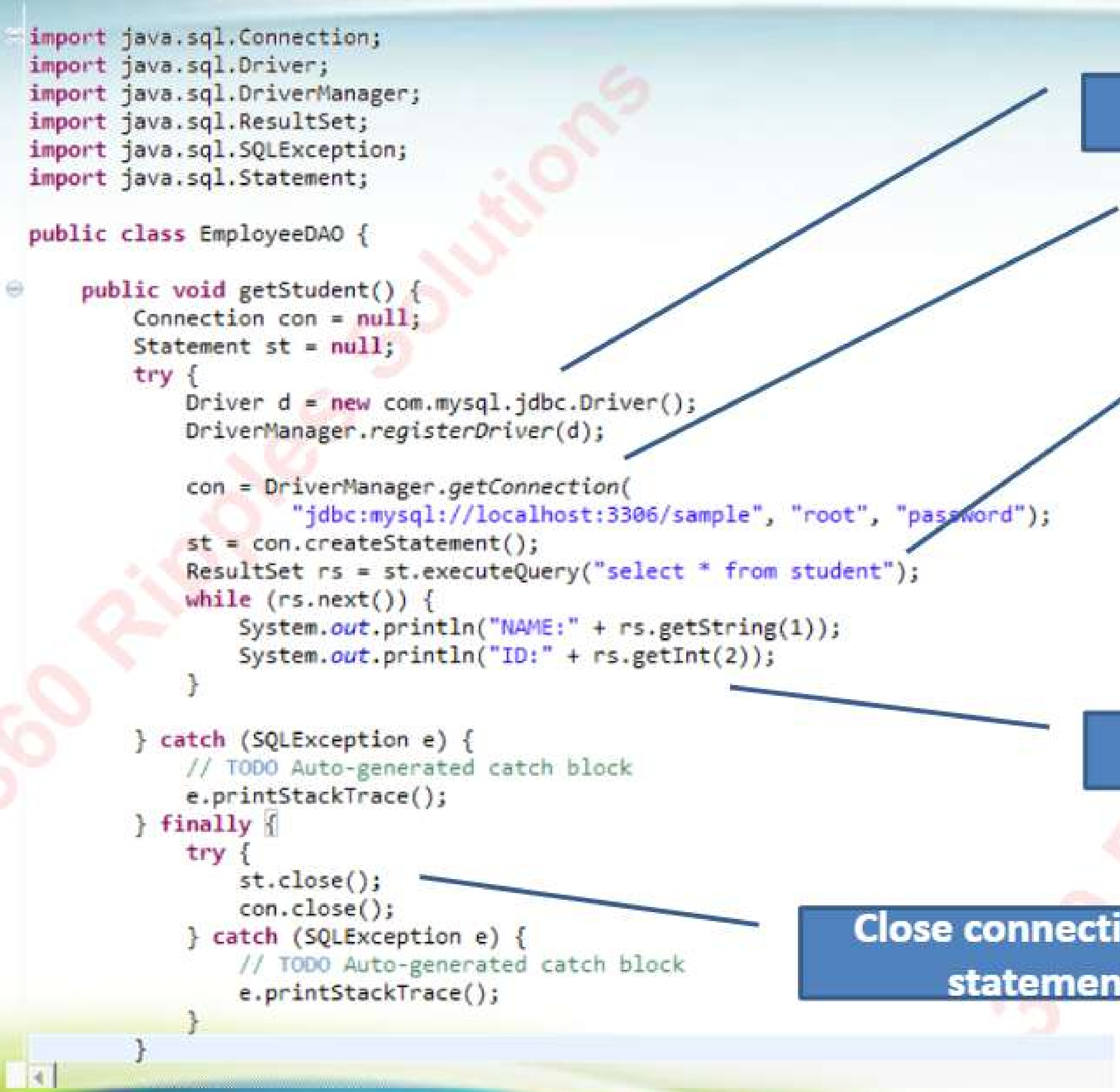
```
rs.close();
stmt.close();
```

9: Closing the Connection

```
conn.close();
```

A quick demo

Click to Continue



```
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class EmployeeDAO {

    public void getStudent() {
        Connection con = null;
        Statement st = null;
        try {
            Driver d = new com.mysql.jdbc.Driver();
            DriverManager.registerDriver(d);

            con = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/sample", "root", "password");
            st = con.createStatement();
            ResultSet rs = st.executeQuery("select * from student");
            while (rs.next()) {
                System.out.println("NAME:" + rs.getString(1));
                System.out.println("ID:" + rs.getInt(2));
            }

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            try {
                st.close();
                con.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

Load the driver

Establish the Connection

Execute query

Iterate records.

Close connection and statements