**Example-3** Program for update the student data using prepared statements.

**package** com.operation;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

//update using prepared statement

**public** **class** UpdateStatement {

**public** **static** **void** main(String[] args) {

**try** {

Class.*forName*("com.mysql.jdbc.Driver"); // load the establish

Connection con = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/sys", "root", "root");

//for single update value

PreparedStatement ps = con.prepareStatement("update employee set username=? where id=?");

// multiple update value

PreparedStatement preparedStatement = con

.prepareStatement("update employee set username=?, password=? where id=?");

ps.setString(1, "ashok");

ps.setString(2, "15");

**int** i = ps.executeUpdate();

System.***out***.println("Record updated." + i);

con.close();

ps.close();

}

**catch** (Exception e) {

e.getMessage();

}

}

}

**Example-4** Program for delete the student data using prepared statements.

package com.operation;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

//delete using prepared statement

public class DeleteStatement {

public static void main(String[] args) {

try {

Class.forName("com.mysql.jdbc.Driver"); // load the establish

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/test", "root", "root");

PreparedStatement ps = con.prepareStatement("delete from employee where id=?");

ps.setInt(1, 8); //here 1 is the parameter index, 8 is the id of table

int i = ps.executeUpdate();

System.out.println("Record deleted." + i);

con.close();

ps.close();

}

catch (Exception e) {

e.getMessage();

}

}

}

**Example-5** Program for retrieve the student data using prepared statements.

**package** com.operation;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**public** **class** SelectStatement {

**public** **static** **void** main(String[] args) {

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection con=DriverManager.*getConnection*("jdbc:mysql://localhost:3306/test","root","root");

//select query

PreparedStatement ps=con.prepareStatement("select \* from employee");

ResultSet rs=ps.executeQuery();

**while**(rs.next()) {

System.***out***.println("ID="+rs.getInt(1));

System.***out***.println("Username="+rs.getString(2));

System.***out***.println("Password="+rs.getString(3));

}

con.close();

ps.close();

rs.close();

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

There are three types of JDBC statements as-

* Statement
* PreparedStatement
* CallableStatement

1. **Statement**

Statement is an interface available in java.sql package

* The statement object can be created using one of the following methods of connection interface:
  + Statement createStatement();
  + Statement createStatement(int,int);
  + Statement createStatement(int,int,int);
* Once the statement object is created, you can call one of the following methods of statement interface:
  + ResultSet executeQuery(String)
  + int executeUpdate(String)
  + boolean execute(String)

1. The executeQuery()method can be used to submit the selected SQL statement to the SQL Engine.  
   This method returns the Resultset object which contains the number of records returned by the given selected SQL statement.
2. The executeUpdate() method can be used to submit insert, update, and delete SQL statement to SQL Engine.  
   This method returns the integer number which represents the number of record affected by the given SQL statement.
3. The execute() method can be used to submit insert, update, delete SQL statement to SQL Engine.  
   This method returns the Boolean value which represents whether the given operation is insert/update/delete (false) OR Fetch (true).
   * Using one statement object, you can submit one or more SQL statements
   * When you submit the SQL statement to SQL Engine using statement object, the SQL statement will be compiled and executed every time.
4. **PreparedStatement**

PreparedStatement is an interface available in java.sql package and it extends the Statement interface.

* The PreparedStatement object can be created using one of the following methods of connection interface:
  + PreparedStatement (String);
  + PreparedStatement (String, int, int);
  + PreparedStatement(String,int,int,int);
* Once the preparedStatement object is created, you can call one of the following methods of preparedStatement interface:
  + ResultSet executeQuery()
  + int executeUpdate()
  + boolean execute()
* Using one preparedStatement object, you can submit only one type of SQL statement.

1. **CollableStatement**

The CallableStatement is an interface available in java.sql package.

The CallableStatement object can be created using one of the following methods of connection interface:

* + CallableStatement preparecall(String);
  + CallableStatement preparecall(String,int,int,);
  + CallableStatement preparecall(String,int,int,int,);
* Once callableStatement object is created, you can call one of the following methods of callableStatement interface:
  + ResultSet executeQuery()
  + int executeUpdate()
  + boolean execute()
* CallableStatement is mainly used to execute stored procedures running in the database.  
  Using one CallableStatement object. You can submit only one call one stored procedure.

Note- When ResultSet record is created initially result set cursor points to before to the first record.

Assignment- Write a program using Callable statements.