Session-5

Callable Statements in JDBC

CallableStatement interface is used to call the stored procedures.

- Therefore, the stored procedure can be called by using an object of the CallableStatement interface.
- The object is created using the prepareCall() method of Connection interface.

```
CallableStatement cs=conn.prepareCall("{call Proc_Name(?,?)}");
cs.setInt(1,2222);

cs.registerOutParameter(2,Types.VARCHAR);
cs.execute();
```

- Three types of parameters exist: IN, OUT, and INOUT.
- PreparedStatement object only uses the IN parameter. The CallableStatement object can use all the three.

Parameter	**Description**
IN	A parameter whose value is unknown when the SQL statement is created. You bind values to IN parameters with the setXXX() methods.
OUT	A parameter whose value is supplied by the SQL statement it returns. You retrieve values from the OUT parameters with the getXXX() methods.
INOUT	A parameter that provides both input and output values. You bind variables with the setXXX() methods and retrieve values with the getXXX() methods.

Example of CallableStatement

Writa a Callable Statement program to retrieve branch of the student using {getBranch() procedure} from given enrollment number. Also write code for Stored Procedure
Stored Procedure: getbranch()

```
DELIMITER @@

DROP PROCEDURE getbranch @@

CREATE PROCEDURE databaseName.getbranch

(IN enr_no INT, OUT my_branch VARCHAR(10))

BEGIN

SELECT branch INTO my_branch

FROM dietStudent

WHERE enr_no=enrno;

END @@

DELIMITER ;
```

Callable Statement program

```
import java.sql.*;

public class CallableDemo {

public static void main(String[] args) {

try {

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

Connection conn= DriverManager.getConnection

("jdbc:mysql://localhost:3306/Diet", "root","pwd"); 8.

CallableStatement cs=conn.prepareCall("{call getbranch(?,?)}");
     cs.setInt(1,2222);

cs.registerOutParameter(2,Types.VARCHAR);

cs.execute();

System.out.println("branch="+cs.getString(2));
cs.close();

conn.close();

}catch(Exceptione){System.out.println(e.toString());}

}//PSVM

}//class
```

Statement, Prepared Statement and Callable Statement :-

Statement	Prepared Statement	Callable Statement
Super interface for Prepared and Callable Statement	extends Statement (sub-interface)	extends PreparedStatement (sub-interface)
Used for executing simple SQL statements like CRUD (create, retrieve, update and delete	Used for executing dynamic and pre-compiled SQL statements	Used for executing stored procedures
The Statement interface cannot accept parameters.	The PreparedStatement interface accepts input parameters at runtime.	The CallableStatement interface can also accept runtime input parameters.
stmt = conn.createStatement();	PreparedStatement ps=con.prepareStatement ("insert into studentDiet values(?,?,?)");	CallableStatement cs=conn.prepareCall("{call getbranch(?,?)}");
java.sql.Statement is slower as compared to Prepared Statement in java JDBC.	PreparedStatement is faster because it is used for executing precompiled SQL statement in java JDBC.	None
java.sql.Statement is suitable for executing DDL commands - CREATE, drop, alter and truncate in java JDBC.	java.sql.PreparedStatement is suitable for executing DML commands - SELECT, INSERT, UPDATE and DELETE in java JDBC.	java.sql.CallableStatement is suitable for executing stored procedure.