# Lab 1: Prepared Statements

## Objective

In this lab we will learn what are prepare statements and how to use them.

## Material

1. Starter file: School.zip
2. Solution file: Lab01-1_PreparedStatements.zip

## Overview

1. Inspect the AdmissionData.txt file
2. Create connection to Database
3. Create and use a prepared statement to create a table for admission data
4. Create and use a prepared statement to add data to admission table
5. Create and use a prepared statement to display admission table data
6. Create and use a prepared statement to update admission table data
7. Create and use a prepared statement to delete admission table data

## Inspect AdmissionData.txt

The admissionData.txt represents data for newly admitted students, the data contain the following student information

- First name
- Last name
- Date of birth
- Gender
- Grade

Determine the appropriate field name, type and size.

| FIRST_NAME | LAST_NAME | DOB | GENDER | GRADE |
|---|---|---|---|---|
| CHAR (25) NOT NULL | CHAR (25) NOT NULL | DATE | CHAR (1) | INT |

## Create connection to database

Create a class called SchoolDatabase.java. In it, create a method to connect to the database.

```
private void databaseConnection() {
		String driverClass = "org.apache.derby.jdbc.EmbeddedDriver";
		String url = "jdbc:derby:memory:school;create=true";
		try {
			Class.forName(driverClass);
			con = DriverManager.getConnection(url);
		}
		catch (ClassNotFoundException e) {
			System.out.println("Class " + driverClass + " could not be found.");
		}
		 catch (SQLException e) {
			System.out.println("SQL Exception in SchoolDatabase.databaseConnection: "+
		e.getMessage());
			e.printStackTrace();
		}
}
```

## Creating Admissions table

Write and execute a statement to create table a called ADMISSIONS.

```
private void createAdmissionsTable() {
		String createSQL = "CREATE TABLE ADMISSIONS (ID int GENERATED ALWAYS AS IDENTITY
		not null primary key,LAST_NAME CHAR(25) NOT NULL, FIRST_NAME CHAR(25) NOT NULL,
		DOB DATE, "
		+ " GENDER CHAR(1), GRADE INT )";
		try {
			stmt = st.executeUpdate(createSQL);
			System.out.println("Table ADMISSIONS created.");
		} catch (SQLException e) {
			System.out.println("SQL Exception in SchoolDAO() createAdmissionsTable() "
			+ e.getMessage());
		}
}
```

## Add data to ADMISSIONS table

Write a method that executes following statement to insert data into the admissions table.

```
String rows = "INSERT INTO ADMISSIONS (LAST_NAME, FIRST_NAME, DOB, GENDER, GRADE) VALUES "+
                    " ( 'Able', 'Adam', '2012-06-1-12.00.00.000000', 'M',1),"+
                    "( 'Baker', 'Betty', '2012-06-1-12.00.00.000000', 'F',2),"+
                    "( 'CASTLE', 'CHARLES', '2012-06-1-12.00.00.000000', 'M',3),"+
                    "( 'DENNING', 'DANIEL', '2012-06-1-12.00.00.000000', 'F',1),"+
                    "( 'Elle', 'Edward', '2012-06-1-12.00.00.000000', 'M',5),"+
                    "( 'Fry', 'Frances', '2012-06-1-12.00.00.000000', 'F',4),"+
                    "( 'GATES', 'GIL', '2012-06-1-12.00.00.000000', 'M',6),"+
                    "( 'Hess', 'Hank', '2012-06-1-12.00.00.000000', 'F',1),"+
                    "( 'Idman', 'Ida', '2012-06-1-12.00.00.000000', 'M',3),"+
                    "( 'JACOBS', 'JAMES', '2012-06-1-12.00.00.000000', 'M',2)";
```

## Create constructor to create and populate the database

Write the constructor to call the previously defined methods to create and populate the database.

```
public SchoolDatabase() {
        databaseConnection();
        createAdmissionsTable();
        loadAdmissionsTable();
}
```

## Create a method to get the connection

## Define a new class

Define a new class called SchoolDAO.java. Create a constructor that takes an instance of the SchoolDatabase and gets the connection from it.

## Display ADMISSION table data

Write a method that executes a prepared statement find and print all records to the console.

```
ResultSet rs = st.executeQuery("SELECT * FROM ADMISSIONS");
while (rs.next()) {
        int id = rs.getInt("ID");
        String firstName = rs.getString("FIRST_NAME").trim();
        String lastName = rs.getString("LAST_NAME").trim();
        String dob = rs.getDate("DOB").toString().trim();
        String gender = rs.getString("GENDER").trim();
        int grade = rs.getInt("GRADE");
        System.out.println(id + "\t" + firstName  + "\t\t"
                + lastName+ "\t" + dob. + "\t"
                + gender + "\t" + grade);
}
```

## Update a record by ID

Write and execute a prepared statement and update FIRST_NAME to 'ALEX' where ID = 3

```
public void updateAdmissionFirstNameById(int id,String firstName) {
        String updateRecord = "UPDATE ADMISSIONS SET FIRST_NAME = ? WHERE ID =?";
        try {
                PreparedStatement stmt = conn.prepareStatement(updateRecord);
                stmt.setString(1, firstName);
                stmt.setInt(2, id);
                st.execute(updateRecord);
        } catch (SQLException e) {
                System.out.println("SQL Exception in SchoolDAO() updateAdmissionFirstNameById()
                        " + e.getMessage());
        }
        System.out.println("Table ADMISSIONS record updated.");
}
```

## Delete a record by ID

Write and execute a prepared statement to delete record from AMISSIONS where ID = 3

```
public void deleteAdmissionById(int id) {
        String deleteRecord = "DELETE FROM ADMISSIONS WHERE ID = ?";
        try {
                PreparedStatement stmt = conn.prepareStatement(deleteRecord);
                stmt.setInt(1, id);
                stmt.execute();
        } catch (SQLException e) {
                System.out.println("SQL Exception in SchoolDAO() deleteAdmissionById() " +
                        e.getMessage());
        }
        System.out.println("Table ADMISSIONS record deleted.");
}
```

## Final result:

Create a class called School.java with main method and execute the code in following sequence using a SchoolDatabase and a SchoolDAO instance

```
public static void main(String args[]) {
        SchoolDatabase db = new SchoolDatabase();
        SchoolDAO dao = new SchoolDAO(db);
        dao.createAdmissionsTable();
        dao.loadAdmissionsTable();
        dao.printAdmissions();
        dao.updateAdmissionFirstNameById(3, "ALEX");
        dao.printAdmissions();
        dao.deleteAdmissionById(3);
        dao.printAdmissions();
```

If everything is in place the outcome in console should be like this

```
Table ADMISSIONS drop dropped.
Table ADMISSIONS created.
Table ADMISSIONS loaded.

ID     FIRST_NAME    LAST_NAME     DOB           GENDER        GRADE
=================================================================================
1      Able          Adam          2012-06-01    M      1
2      Baker         Betty         2012-06-01    F      2
3      CASTLE        CHARLES       2012-06-01    M      3
4      DENNING       DANIEL        2012-06-01    F      1
5      Elle          Edward        2012-06-01    M      5
6      Fry           Frances       2012-06-01    F      4
7      GATES         GIL           2012-06-01    M      6
```

```
8      Hess         Hank         2012-06-01   F      1
9      Idman        Ida          2012-06-01   M      3
10     JACOBS       JAMES        2012-06-01   M      2


Total Item Count: 10

Table ADMISSIONS record updated.

ID      FIRST_NAME      LAST_NAME       DOB             GENDER  GRADE
=================================================================================
1       Able            Adam            2012-06-01      M       1
2       Baker           Betty           2012-06-01      F       2
3       ALEX            CHARLES 2012-06-01      M       3
4       DENNING DANIEL          2012-06-01      F       1
5       Elle            Edward          2012-06-01      M       5
6       Fry             Frances 2012-06-01      F       4
7       GATES           GIL             2012-06-01      M       6
8       Hess            Hank            2012-06-01      F       1
9       Idman           Ida             2012-06-01      M       3
10      JACOBS          JAMES           2012-06-01      M       2


Total Item Count: 10

Table ADMISSIONS record deleted.

ID      FIRST_NAME      LAST_NAME       DOB             GENDER  GRADE
=================================================================================
1       Able            Adam            2012-06-01      M       1
2       Baker           Betty           2012-06-01      F       2
4       DENNING DANIEL          2012-06-01      F       1
5       Elle            Edward          2012-06-01      M       5
6       Fry             Frances 2012-06-01      F       4
7       GATES           GIL             2012-06-01      M       6
8       Hess            Hank            2012-06-01      F       1
9       Idman           Ida             2012-06-01      M       3
10      JACOBS          JAMES           2012-06-01      M       2


Total Item Count: 9
```