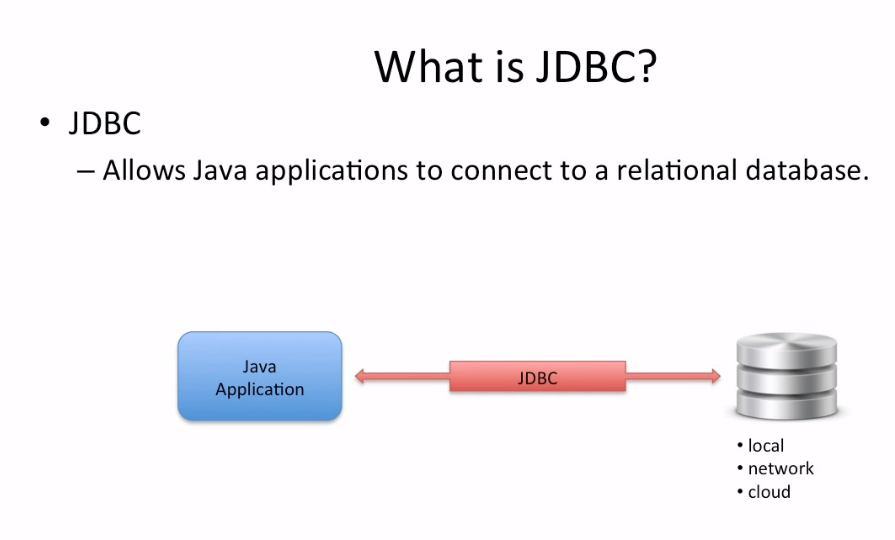
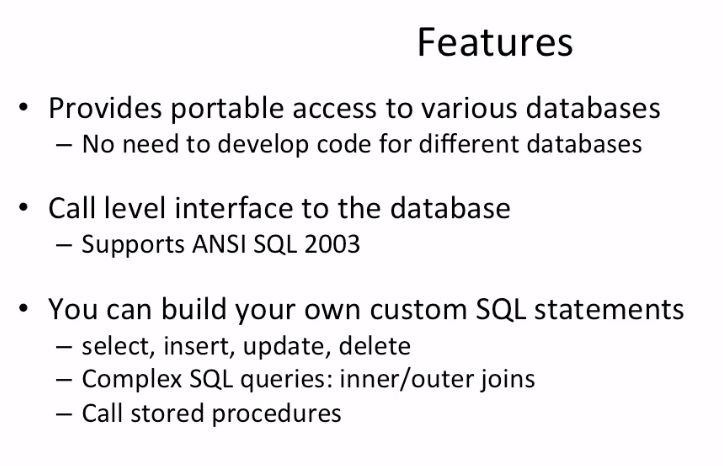
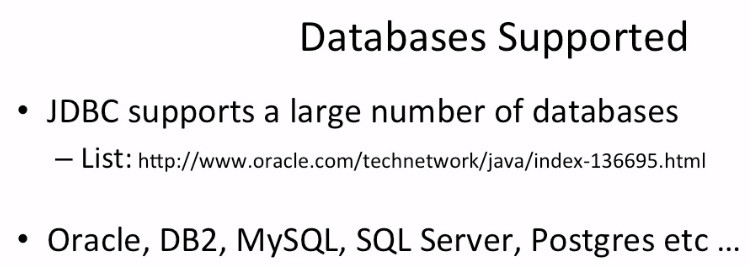
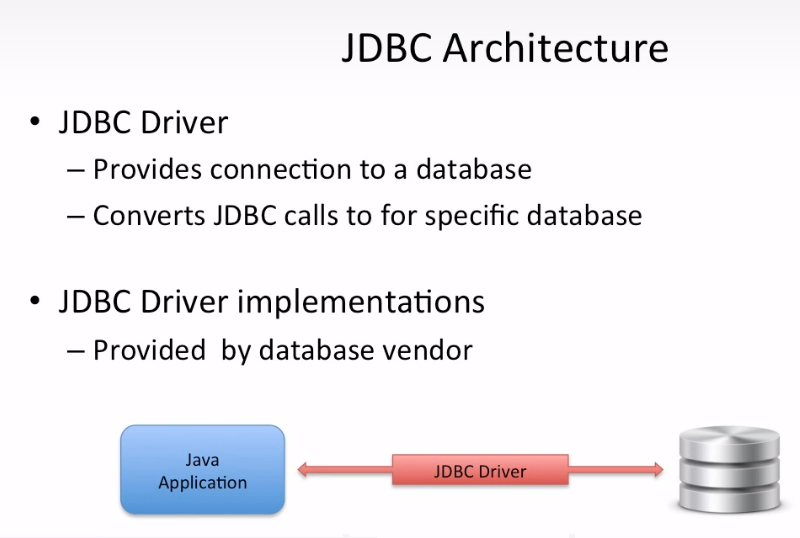
Section 02: JDBC Overview

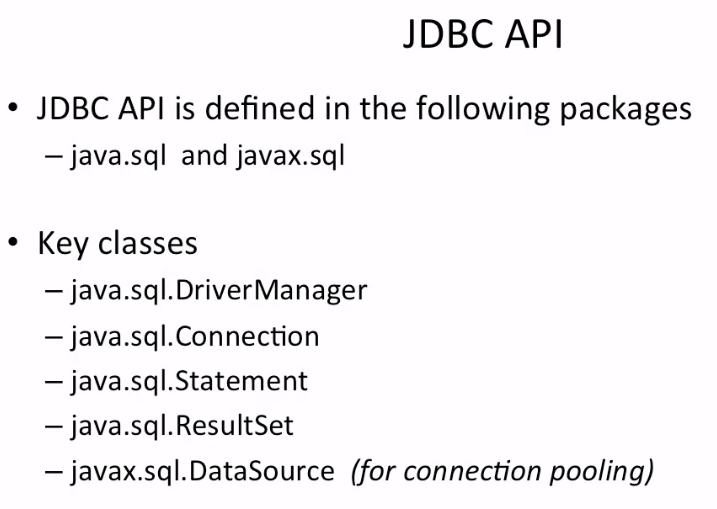




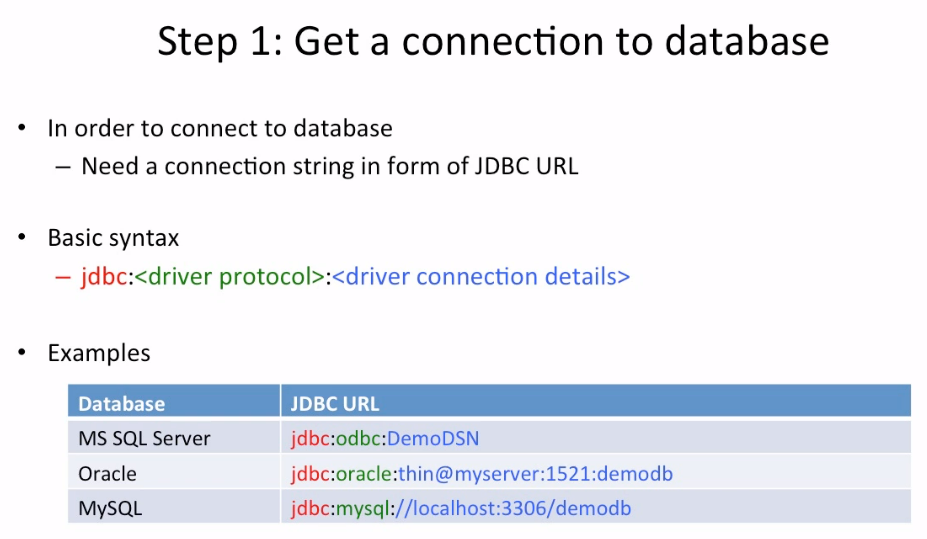


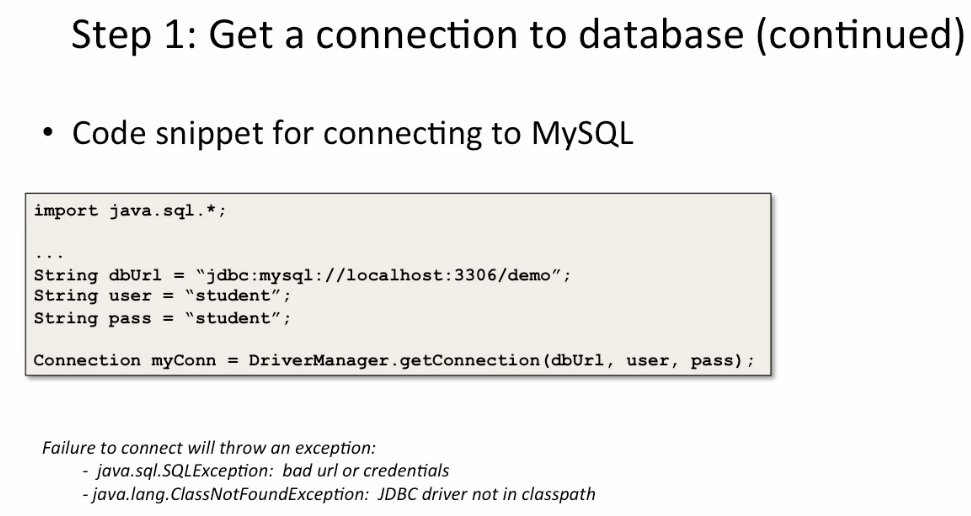


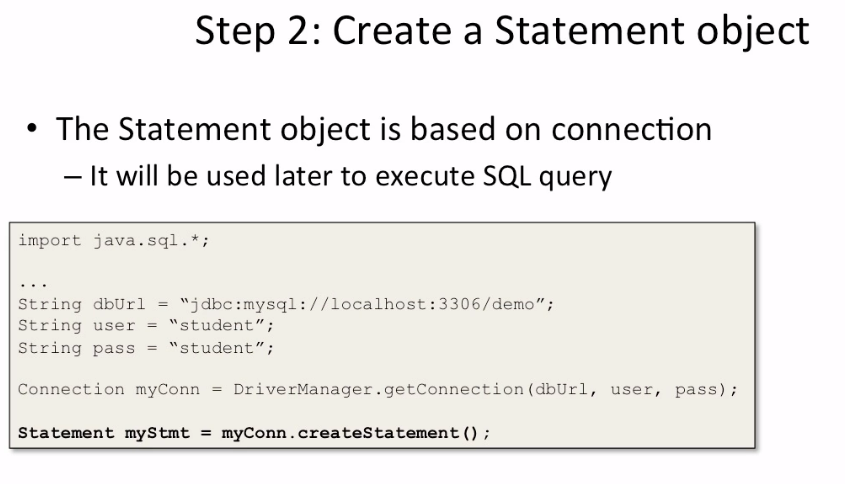


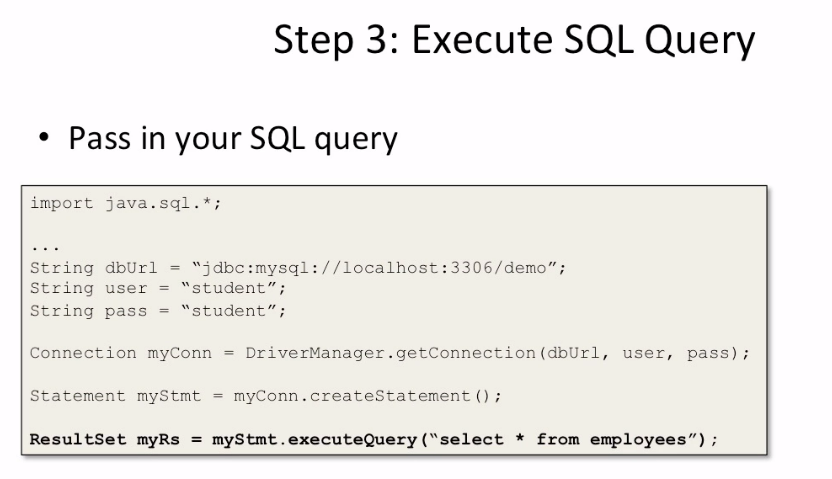


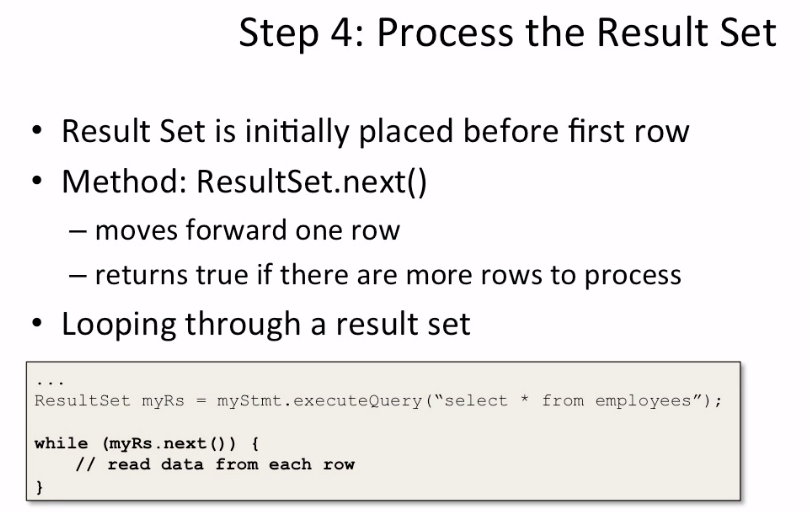


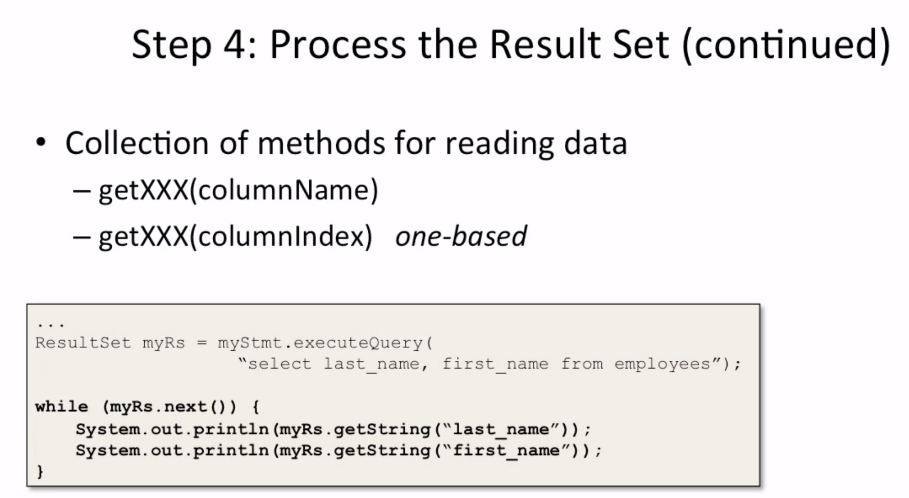


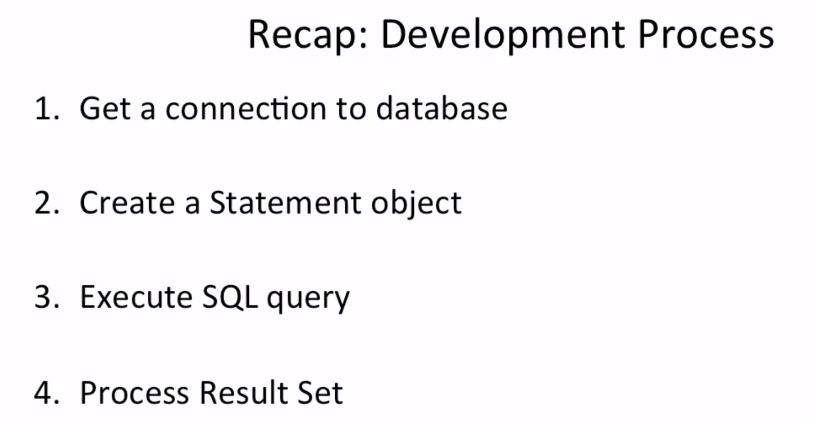




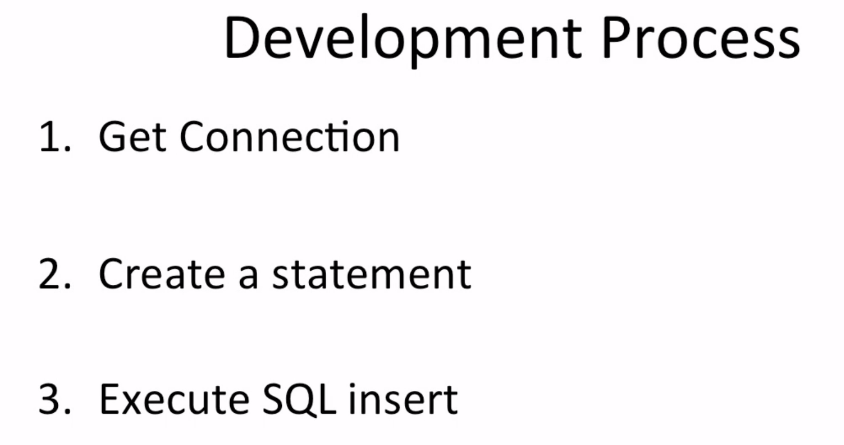








Inserting data into the database:



This generic instruction:

myStmt.executeUpdate()

allows is used to perform inserts, updates and deletes

int rowsAffected = myStmt.executeUpdate(  
 "insert into employees " +  
 "(last\_name, first\_name, email, department, salary) " +  
 "values " +  
 "('Wright', 'Eric', 'eric.wright@foo.com', 'HR', 33000.00)");

this method returns the number of rows affected, one in this case and we use the variable rowsAffected to save this value

**Difference between Statement and PreparedStatement:**

**1. Statement :**   
It is used for accessing your database. Statement interface cannot accept parameters and useful when you are using static SQL statements at runtime. If you want to run [**SQL**](https://www.geeksforgeeks.org/sql-tutorial/)**query only once** then this interface is preferred over PreparedStatement.

**Example –**

//Creating The Statement Object

Statement GFG = con.createStatement();

//Executing The Statement

GFG.executeUpdate("CREATE TABLE STUDENT(ID NUMBER NOT NULL, NAME VARCHAR)");

**2. PreparedStatement :**   
It is used when you want to use **SQL statements many times.** The PreparedStatement interface accepts input parameters at runtime. 

**Example –**

//Creating the PreparedStatement object

PreparedStatement GFG = con.prepareStatement("update STUDENT set NAME = ? where ID = ?");

//Setting values to place holders

//Assigns "RAM" to first place holder

GFG.setString(1, "RAM");

//Assigns "512" to second place holder

GFG.setInt(2, 512);

//Executing PreparedStatement

GFG.executeUpdate();

**Table: Difference between Statement and PreparedStatement :** 

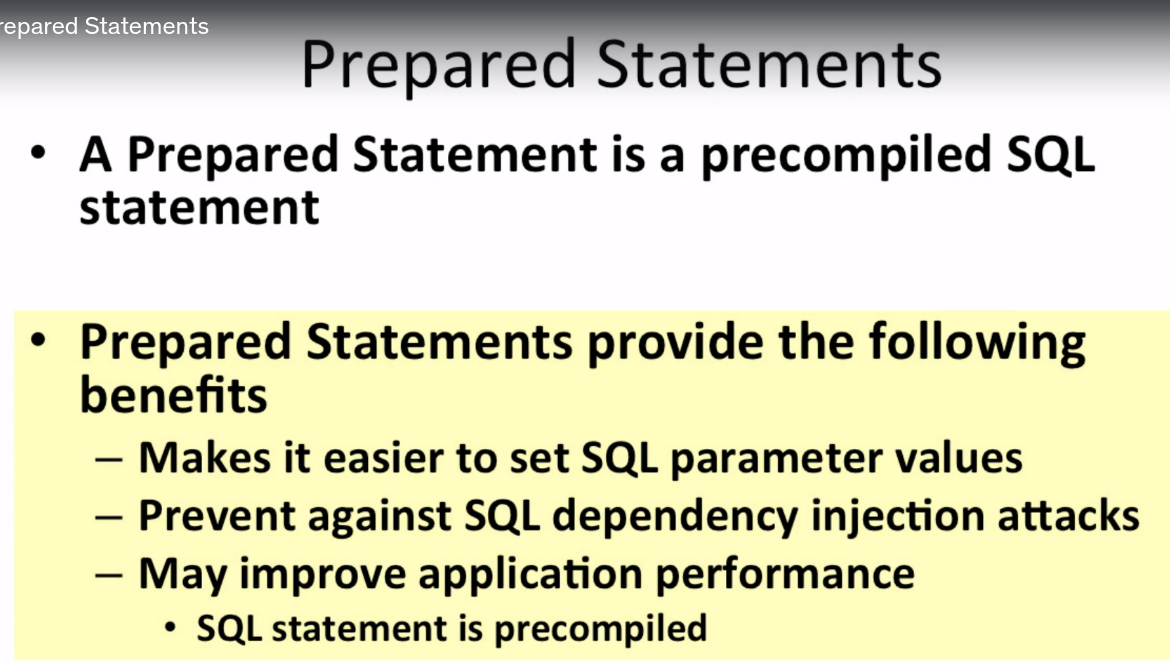
|  |  |
| --- | --- |
| Statement | PreparedStatement |
| * It is used when SQL query is to be executed only once. | * It is used when SQL query is to be executed multiple times. |
| * You cannot pass parameters at runtime. | * You can pass parameters at runtime. |
| * Used for CREATE, ALTER, DROP statements. | * Used for the queries which are to be executed multiple times. |
| * Performance is very low. | * Performance is better than Statement. |
| * It is base interface. | * It extends statement interface. |
| * Used to execute normal SQL queries. | * Used to execute dynamic SQL queries. |
| * We cannot use statement for reading binary data. | * We can use Preparedstatement for reading binary data. |
| * It is used for DDL statements. | * It is used for any SQL Query. |
| * We cannot use statement for writing binary data. | * We can use Preparedstatement for writing binary data. |
| * No binary protocol is used for communication. | * Binary protocol is used for communication. |

Prepared Statement:

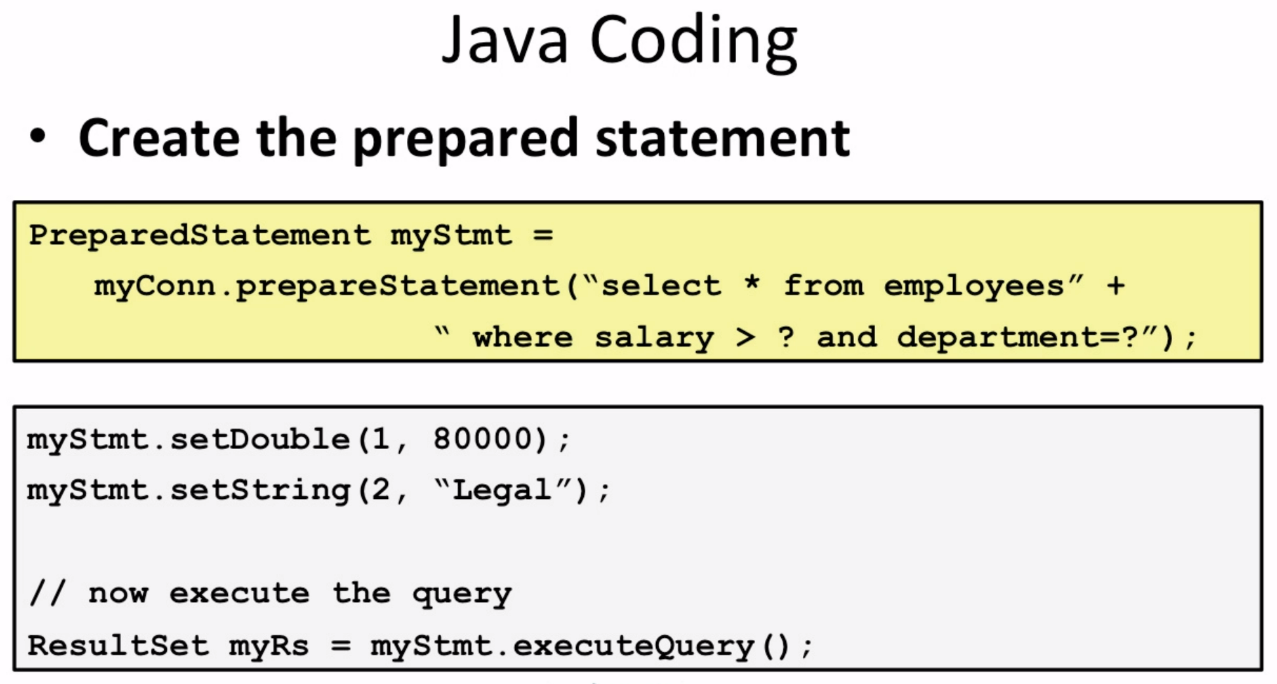
A **PreparedStatement** is a pre-compiled SQL statement. It is a subinterface of **Statement**. Prepared Statement objects have some useful additional features than Statement objects. Instead of hard coding queries, PreparedStatement object provides a feature to execute a parameterized query.

**Advantages of PreparedStatement**

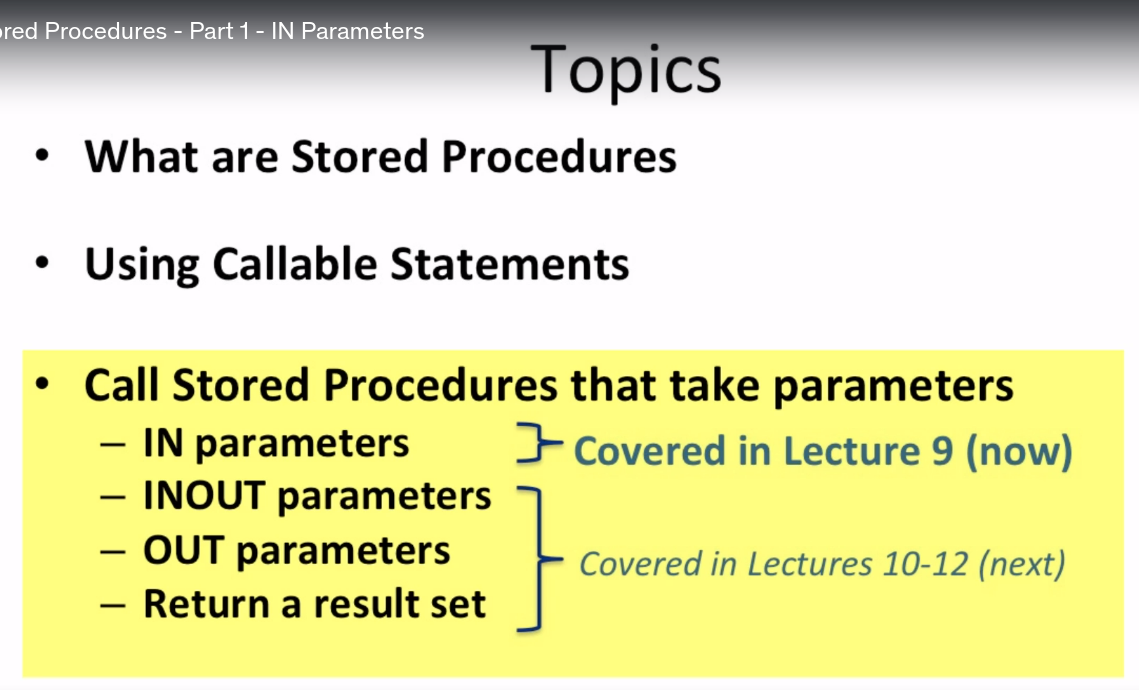
* When PreparedStatement is created, the SQL query is passed as a parameter. This Prepared Statement contains a pre-compiled SQL query, so when the PreparedStatement is executed, DBMS can just run the query instead of first compiling it.
* We can use the same PreparedStatement and supply with different parameters at the time of execution.
* An important advantage of PreparedStatements is that they prevent SQL injection attacks.

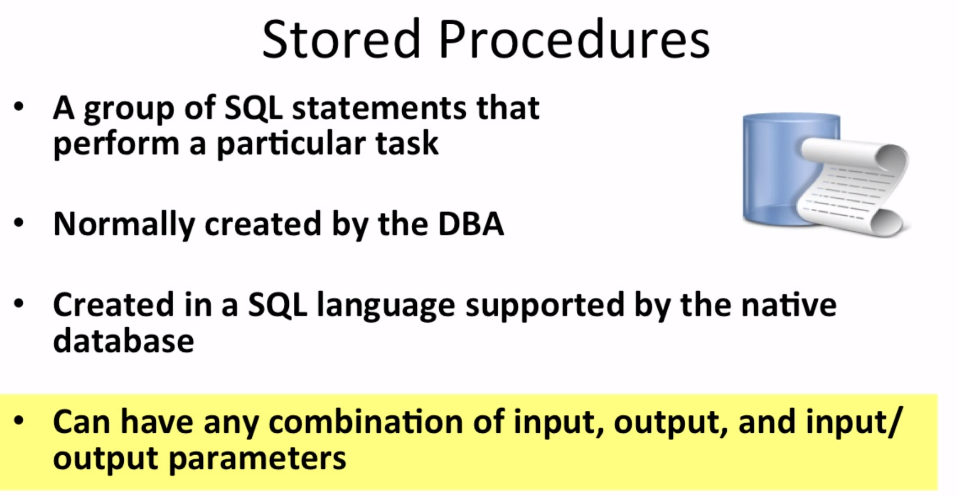


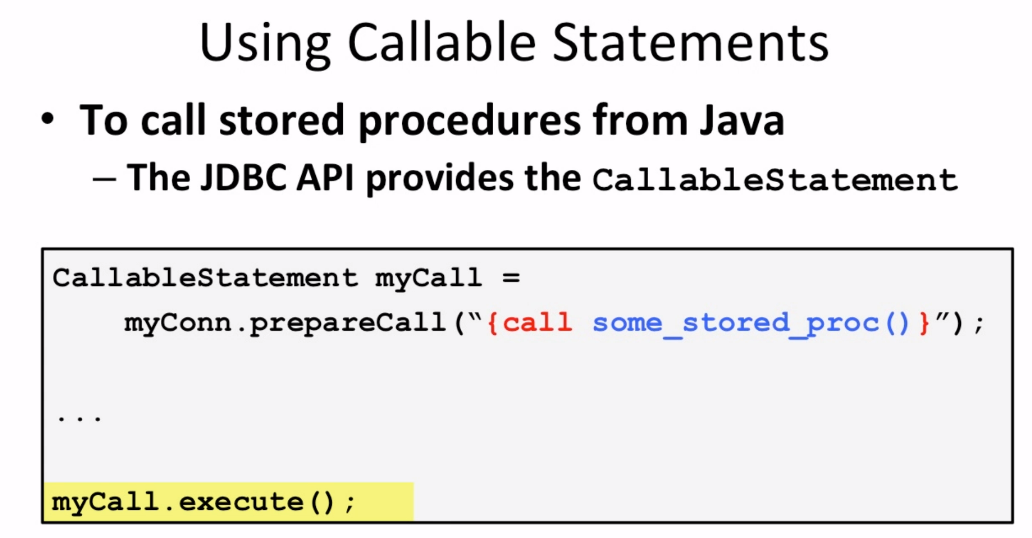


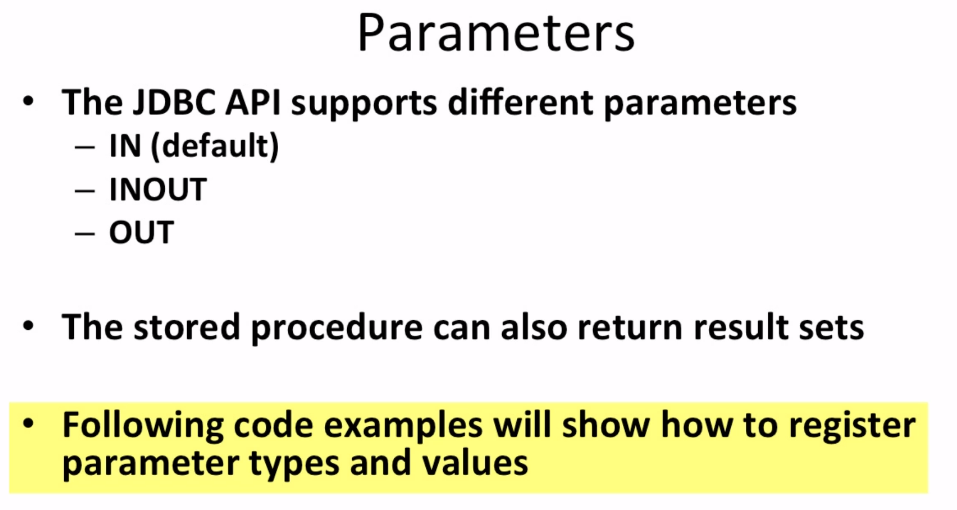


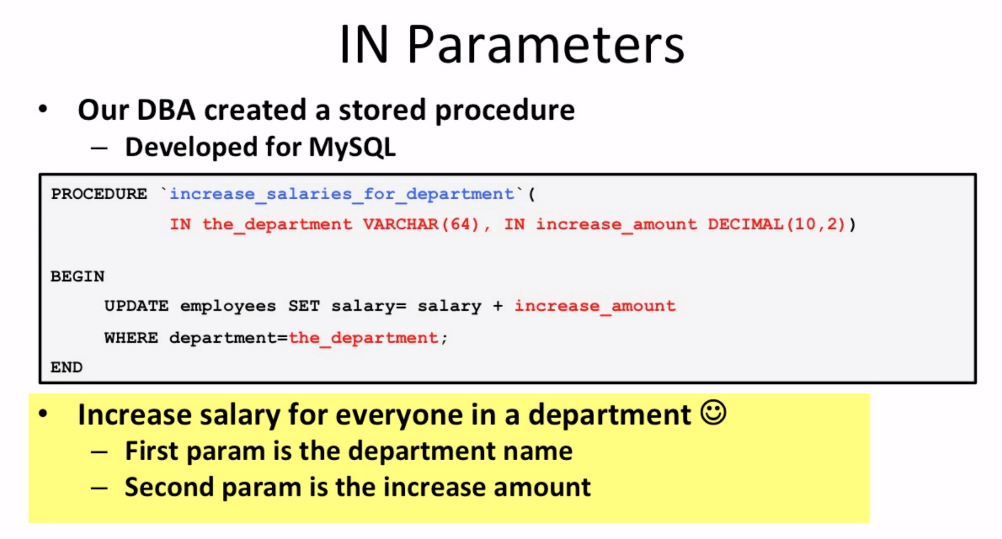
**Stored Procedures:**

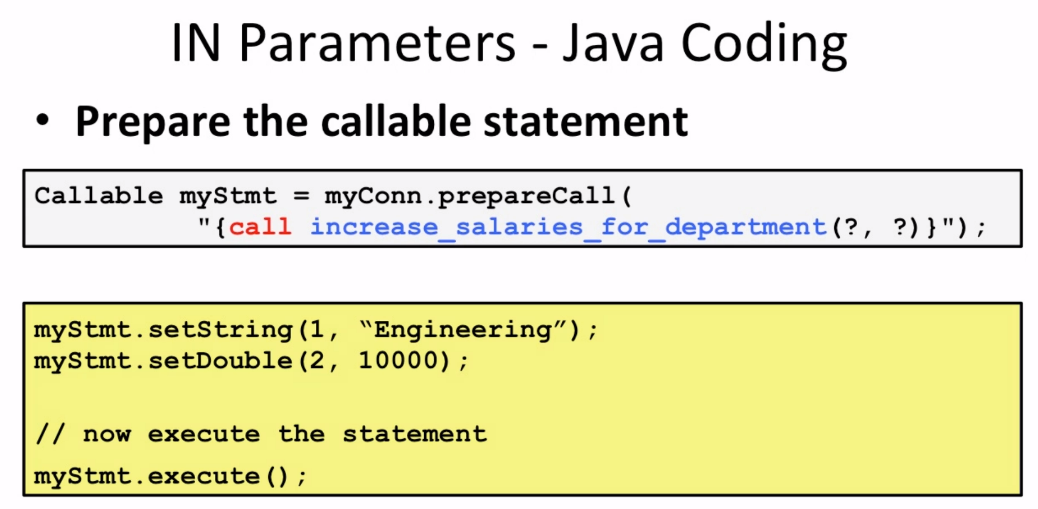




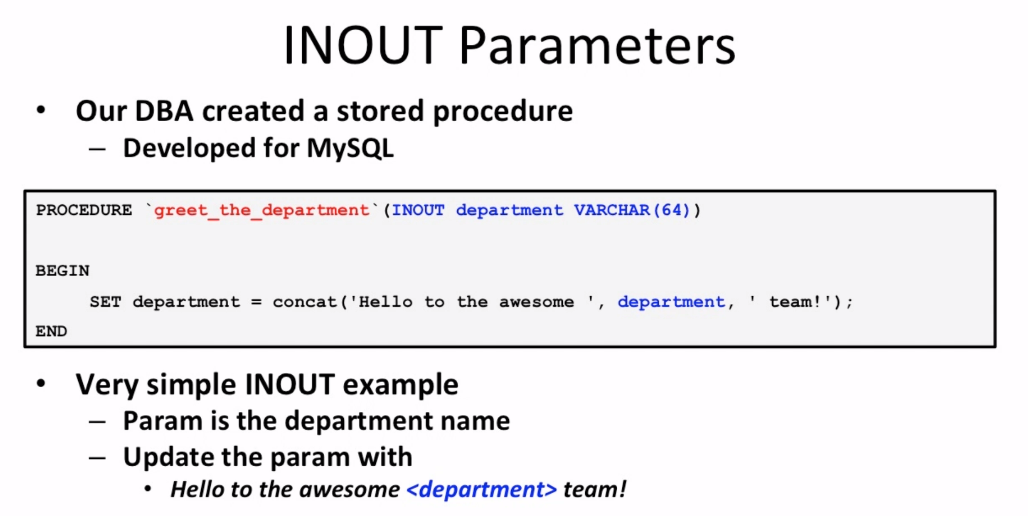


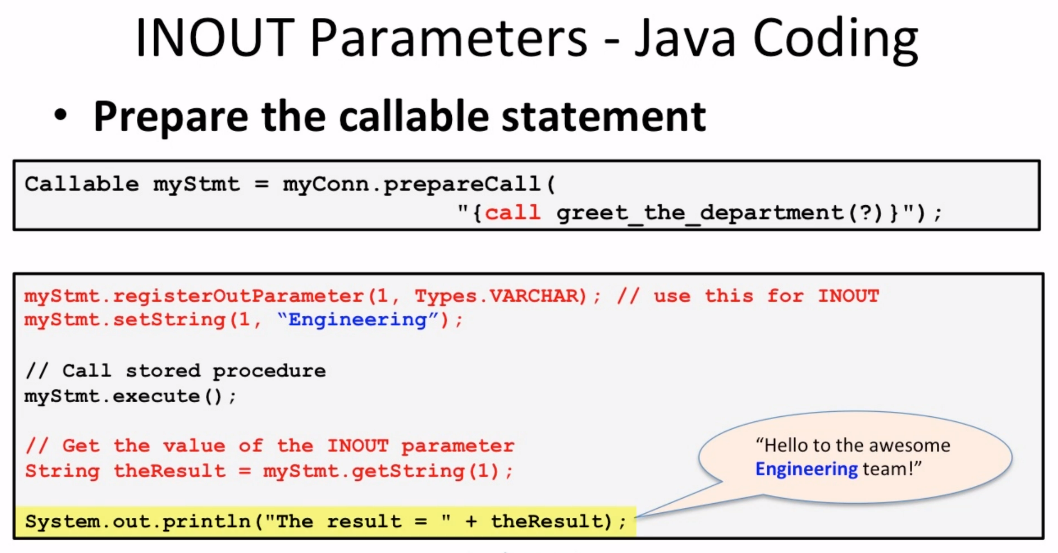






INOUT parameters:





OUT parameters:



