

RIGA TECHNICAL UNIVERSITY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

INSTITUTE OF APPLIED COMPUTER SYSTEMS

“Technology of Large database”

Practical task 2#

**Use of Java in database server**

Author: Batuhan Arslanoglu

Studentcardno.: 151ADB096

2017 / 2018 study year

Contents

[1. Task description 4](#_Toc505898279)

[2. Java class creation and loading in database 4](#_Toc505898280)

[2.1 Using LOADJAVA program 4](#_Toc505898281)

[a) Creation of Java class 4](#_Toc505898282)

[b) Compiling the java class by using the standard Java compiler: 5](#_Toc505898283)

[c) Here, I will show the loading java class into server by using Loadjava command: 5](#_Toc505898284)

[d) PL/SQL procedure: 5](#_Toc505898285)

[e) Publish the stored procedure through a call specification 6](#_Toc505898286)

[2.2 Using CREATE JAVA command 6](#_Toc505898287)

[a) PL/SQL procedure: 7](#_Toc505898288)

[b) Publish the stored procedure through a call specification 7](#_Toc505898289)

[c) Metadata in SQL developer 8](#_Toc505898290)

[3. Java class meta-date inspection (SELECT) 8](#_Toc505898291)

[a) Java class 8](#_Toc505898292)

[b) Loading java class into server 9](#_Toc505898293)

[c) Function creation for calculating the balance: 9](#_Toc505898294)

[d) Creation of object type and MEMBER methods: 10](#_Toc505898295)

[e) Table creation: 10](#_Toc505898296)

[f) Output of data 11](#_Toc505898297)

[4. JDBC stored procedures 11](#_Toc505898298)

[4.1 Stored procedures 11](#_Toc505898299)

[a) Creation of tables for stored procedures and JDBC 11](#_Toc505898300)

[b) Stored procedure with input and output 12](#_Toc505898301)

[c) Stored procedure Cursor: 12](#_Toc505898302)

[4.2 JDBC - CallableStatement In Out Parameters 13](#_Toc505898303)

[a) Creation of procedure for using with JDBC: 13](#_Toc505898304)

[b) JDBC code in Eclipse: 14](#_Toc505898305)

[c) Output from Java class: 16](#_Toc505898306)

[4.3 JDBC - CallableStatement Cursor(Oracle) 17](#_Toc505898307)

[Stored procedure: 17](#_Toc505898308)

[b) JDBC code in Eclipse to get all data from table via stored procedure: 17](#_Toc505898309)

[c) Output from Java class in Eclipse: 20](#_Toc505898310)

[5. Inserting Data from Java class (JDBC): 20](#_Toc505898311)

[a) Creation of Student table: 20](#_Toc505898312)

[b) JDBC code: 20](#_Toc505898313)

[c) Output of data with select to see the data 22](#_Toc505898314)

[6. Select statement from Java method: 22](#_Toc505898315)

[a) Java code: 22](#_Toc505898316)

[b) Output: 23](#_Toc505898317)

[7. Conclusion 23](#_Toc505898318)

# Task description

1. Java class creation and loading in database:  
1) using LOADJAVA program;  
2) using CREATE JAVA command.  
2. Java class meta-date inspection (SELECT).  
3. Creation of PL/SQL function which call Java method. Execution of Java method.  
4. Java class creation, loading and testing for SELECT and INSERT queries realization.  
5. Conclusions (what seems good, what bad, what like, what is problematic).

# Java class creation and loading in database

## 2.1 Using LOADJAVA program

### a) Creation of Java class

|  |
| --- |
| **public** **class** Bookstore {    **public** **static** **final** **int** ***ID***[] = {1, 2, 3};  **public** **static** **final** String ***book\_name***[] = {"PL/SQL", "JDBC", "JavaScript"};  **public** **static** **final** **int** ***year***[] = {2011, 2018, 2015};  **public** **static** **final** String ***category***[] = {"Science", "Science", "Science"};    **public** **static** String Book() {    **int** id = 2;  **switch**(id) {  **case** 1:  **return** ***book\_name***[1] + " is from "+ ***category***[1] + ", published in " + ***year***[1];  **case** 2:  **return** ***book\_name***[2] + " is from " + ***category***[2] + ", published in " + ***year***[2];  **case** 3:  **return** ***book\_name***[3] + " is from " + ***category***[3] + ", published in " + ***year***[3];  **default**:  **return** "Please, enter valid id";  }    }  **public** **static** **void** main(String[] args) {  System.***out***.println(*Book*());    }    } |

### b) Compiling the java class by using the standard Java compiler:

|  |
| --- |
| **C:\Java\_for\_database>javac Bookstore.java**  A screenshot of a cell phone  Description generated with very high confidence |

### c) Here, I will show the loading java class into server by using Loadjava command:

|  |
| --- |
| C:\Java\_for\_database>**loadjava -u HR/hr@localhost:1521/pdborcl -v -r -t Bookstore.java**  A screenshot of a computer  Description generated with very high confidence  We can also see the file in SQL developer:  A screenshot of a cell phone  Description generated with very high confidence |

### d) PL/SQL procedure:

The class file is now loaded into the database and visible from the dba\_objects view with an object type of JAVA CLASS.   
  
From SQL\*Plus, create the PL/SQL wrapper to invoke the newly loaded Java class:

|  |
| --- |
| CREATE OR REPLACE FUNCTION Book RETURN VARCHAR2 AS  LANGUAGE JAVA NAME 'Bookstore.Book() return java.lang.String';  / |

### e) Publish the stored procedure through a call specification

|  |
| --- |
| VARIABLE myString VARCHAR2(100);  CALL Book() INTO :myString;    PRINT myString;  A screenshot of a cell phone  Description generated with very high confidence |

## 2.2 Using CREATE JAVA command

I used Eclipse IDE for creation of Java class and now I allocating Java code from Eclipse so that it becomes visible and readable easily:

|  |
| --- |
| CREATE or REPLACE JAVA SOURCE NAMED "Billionaire" AS  **public** **class** Employee {    **public** **static** **final** **int** ***ID***[] = {1, 2, 3};  **public** **static** **final** String ***name***[] = {"Janis", "Merry", "John"};  **public** **static** **final** **int** ***year\_experience***[] = {5, 10, 15};  **public** **static** **final** String ***position***[] = {"Manager", "Developer", "Accountant"};    **public** **static** String Employee() {    **int** id = 2;  **switch**(id) {  **case** 1:  **return** ***name***[0] + " works as "+ ***position***[0] + " for " + ***year\_experience***[0];  **case** 2:  **return** ***name***[1] + " works as "+ ***position***[1] + " for " + ***year\_experience***[1];  **case** 3:  **return** ***name***[2] + " works as "+ ***position***[2] + " for " + ***year\_experience***[2];  **default**:  **return** "Please, enter valid id";  }  }  **public** **static** **void** main(String[] args) {    System.***out***.println(*Employee*());  }  }  /  **A screen shot of a computer  Description generated with high confidence** |

### PL/SQL procedure:

The class file is now loaded into the database and visible from the dba\_objects view with an object type of JAVA CLASS.   
  
From SQL\*Plus, create the PL/SQL wrapper to invoke the newly loaded Java class:

|  |
| --- |
| CREATE OR REPLACE FUNCTION Employee RETURN VARCHAR2 AS  LANGUAGE JAVA NAME 'Employee.Employee() return java.lang.String';  /  A screenshot of a cell phone  Description generated with very high confidence |

### Publish the stored procedure through a call specification

|  |
| --- |
| VARIABLE myString2 VARCHAR2(100);  CALL Employee() INTO :myString2;    PRINT myString2;  A screenshot of a cell phone  Description generated with very high confidence |

### Metadata in SQL developer

|  |
| --- |
| select OBJECT\_NAME, OBJECT\_TYPE, STATUS, CREATED,  GENERATED from USER\_OBJECTS where CREATED >= TO\_DATE('08-02-2018', 'DD-MM-YYYY');  A screenshot of a cell phone  Description generated with very high confidence |

# Java class meta-date inspection (SELECT)

Here, I am going to show usage of java class with Object Member methods:

### Java class

|  |
| --- |
| **import** java.sql.\*;  **import** oracle.jdbc.\*;  **import** java.util.\*;  **import** java.io.\*;  **public** **class** Calculator {  **public** **static** **int** Calculate(**int** balance, **int** expenses, **int** salary)  **throws** SQLException{  **return** balance+salary-expenses;  }  } |

### b) Loading java class into server

|  |
| --- |
| **C:\Java\_for\_database>loadjava -u** [**HR/hr@localhost:1521/pdborcl -v -r -t Calculator.java**](mailto:HR/hr@localhost:1521/pdborcl%20-v%20-r%20-t%20Calculator.java)  **A screenshot of a computer  Description generated with very high confidence** |

### c) Function creation for calculating the balance:

|  |
| --- |
| CREATE OR REPLACE FUNCTION CAL\_FUNCTION(CURRENT\_BALANCE IN NUMBER, EXPENSES IN NUMBER,  SALARY IN NUMBER) RETURN NUMBER  AS LANGUAGE JAVA NAME 'Calculator.Calculate(java.lang.Integer, java.lang.Integer, java.lang.Integer)  return java.lang.int';  /  A close up of a black background  Description generated with high confidence |

### Creation of object type and MEMBER methods:

|  |
| --- |
| CREATE OR REPLACE BALANCE AS OBJECT (  ID NUMBER,  CURRENT\_BLANCE NUMBER,  EXPENSES NUMBER,  SALARY NUMBER,  MEMBER FUNCTION CALCULATOR RETURN NUMBER);  /  A screenshot of a cell phone  Description generated with very high confidence  CREATE OR REPLACE TYPE BODY BALANCE AS  MEMBER FUNCTION CALCULATOR RETURN NUMBER IS  BEGIN  RETURN (CAL\_FUNCTION(SELF.CURRENT\_BLANCE, SELF.EXPENSES, SELF.SALARY));  END CALCULATOR;  END;  /  A screenshot of a cell phone  Description generated with very high confidence |

### Table creation:

|  |
| --- |
| CREATE TABLE BALANCE\_TAB OF BALANCE;  A screenshot of a cell phone  Description generated with very high confidence |

### Output of data

|  |
| --- |
| INSERT INTO BALANCE\_TAB VALUES(1, 1000, 200, 540);  INSERT INTO BALANCE\_TAB VALUES(2, 800, 200, 250);  INSERT INTO BALANCE\_TAB VALUES(3, 700, 180, 800);  INSERT INTO BALANCE\_TAB VALUES(4, 900, 150, 600);  A screenshot of a cell phone screen with text  Description generated with very high confidence |

# JDBC stored procedures

Here, I am going to show how I create the stored procedure and its usage with Java class

## 4.1 Stored procedures

### a) Creation of tables for stored procedures and JDBC

|  |
| --- |
| CREATE TABLE COUNTRIES(  COUNTRY\_ID NUMBER,  COUNTRY\_NAME VARCHAR2(30),  REGION\_ID NUMBER  );  INSERT INTO COUNTRIES VALUES(101, 'GERMANY', 44);  INSERT INTO COUNTRIES VALUES(102, 'LATVIA', 371);  INSERT INTO COUNTRIES VALUES(103, 'TURKEY', 999);  INSERT INTO COUNTRIES VALUES(104, 'RUSSIA', 78);  A screenshot of a cell phone  Description generated with very high confidence |

### b) Stored procedure with input and output

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GET\_COUNTRY\_NAME  (  COUNTRY\_ID\_INPUT IN NUMBER  , COUNTRY\_NAME\_OUTPUT OUT VARCHAR2  ) AS  BEGIN  SELECT COUNTRY\_NAME INTO COUNTRY\_NAME\_OUTPUT FROM COUNTRIES WHERE COUNTRY\_ID=COUNTRY\_ID\_INPUT;  END GET\_COUNTRY\_NAME;  **QUERY\_1#:**  DECLARE  COUNTRY\_ID\_INPUT NUMBER;  COUNTRY\_NAME\_OUTPUT VARCHAR2(200);  BEGIN  COUNTRY\_ID\_INPUT := 102;  GET\_COUNTRY\_NAME(  COUNTRY\_ID\_INPUT => COUNTRY\_ID\_INPUT,  COUNTRY\_NAME\_OUTPUT => COUNTRY\_NAME\_OUTPUT  );  /\* Legacy output:  DBMS\_OUTPUT.PUT\_LINE('COUNTRY\_NAME\_OUTPUT = ' || COUNTRY\_NAME\_OUTPUT);  \*/  :COUNTRY\_NAME\_OUTPUT := COUNTRY\_NAME\_OUTPUT;  END;  A screenshot of a cell phone  Description generated with very high confidence |

### c) Stored procedure Cursor:

|  |
| --- |
| create or replace PROCEDURE GET\_ALL\_COUNTRIES  (  COUNTRY\_CURSOR OUT SYS\_REFCURSOR  ) AS  BEGIN  OPEN COUNTRY\_CURSOR FOR  SELECT \* FROM COUNTRIES;  END GET\_ALL\_COUNTRIES;  A screenshot of a cell phone  Description generated with very high confidence  **QUERY\_1#:**  create or replace  PROCEDURE GET\_ALL\_COUNTRIES  (  COUNTRY\_CURSOR OUT SYS\_REFCURSOR  ) AS  BEGIN  OPEN COUNTRY\_CURSOR FOR  SELECT \* FROM COUNTRIES;  END GET\_ALL\_COUNTRIES;  A screenshot of a cell phone  Description generated with very high confidence |

## 4.2 JDBC - CallableStatement In Out Parameters

### a) Creation of procedure for using with JDBC:

|  |
| --- |
| create or replace PROCEDURE GET\_COUNTRIES  (  COUNTRY\_CURSOR OUT SYS\_REFCURSOR,  REGION\_ID\_IN IN NUMBER,  COUNTRY\_ID\_OUT OUT VARCHAR2,  COUNTRY\_NAME\_OUT OUT VARCHAR2,  REGION\_ID\_OUT OUT VARCHAR2  ) AS  BEGIN  OPEN COUNTRY\_CURSOR FOR  SELECT COUNTRY\_ID,COUNTRY\_NAME,REGION\_ID INTO COUNTRY\_ID\_OUT,COUNTRY\_NAME\_OUT,REGION\_ID\_OUT  FROM COUNTRIES WHERE REGION\_ID=REGION\_ID\_IN;  END GET\_COUNTRIES;  A screenshot of a cell phone  Description generated with very high confidence |

### b) JDBC code in Eclipse:

|  |
| --- |
| **import** java.sql.CallableStatement;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.ResultSet;  **import** java.sql.SQLException;  **import** java.util.Scanner;  **import** oracle.jdbc.OracleTypes;  **public** **class** JDBCCallable  {  // JDBC driver name and database URL  **static** **final** String ***JDBC\_DRIVER*** = "oracle.jdbc.driver.OracleDriver";  **static** **final** String ***DB\_URL*** = "jdbc:oracle:thin:@localhost:1521:orcl";  // Database credentials  **static** **final** String ***USERNAME*** = "system";  **static** **final** String ***PASSWORD*** = "123456789";  **public** **static** **void** main(String[] args)  {  JDBCCallable jdbcCallableStatementDemo = **new** JDBCCallable();  Scanner scanner = **new** Scanner(System.***in***);  **while** (**true**)  {  System.***out***.print("Enter the RegionId :");  **int** regionId = scanner.nextInt();  **if** (regionId == 0)  {  **break**;  }  jdbcCallableStatementDemo.getCountryInformation(regionId);  }  scanner.close();  }  **private** **void** getCountryInformation(**int** regionId)  {  Connection connection = **null**;  CallableStatement callableStatement = **null**;  **try**  {  /\*  \* Register the JDBC driver in DriverManager  \*/  Class.*forName*(***JDBC\_DRIVER***);  /\*  \* Establish connection to the Database using DriverManager  \*/  connection = DriverManager  .*getConnection*(***DB\_URL***, ***USERNAME***, ***PASSWORD***);  String plSql = "{call GET\_COUNTRIES(?,?,?,?,?)}";  callableStatement = connection.prepareCall(plSql);  /\*  \* Bind IN and OUT parameters  \*/  callableStatement.registerOutParameter(1, OracleTypes.***CURSOR***);  callableStatement.setInt(2, regionId);  callableStatement.registerOutParameter(3, java.sql.Types.***VARCHAR***);  callableStatement.registerOutParameter(4, java.sql.Types.***VARCHAR***);  callableStatement.registerOutParameter(5, java.sql.Types.***INTEGER***);  /\*  \* Use execute method to run the stored procedure.  \*/  callableStatement.executeQuery();  ResultSet rs = (ResultSet) callableStatement.getObject(1);  **while** (rs.next())  {  System.***out***.print("COUNTRY\_ID : " + rs.getString(1));  System.***out***.print(", COUNTRY\_NAME : " + rs.getString(2));  System.***out***.println(", REGION\_ID : " + rs.getString(3));  }  }  **catch** (SQLException se)  {  se.printStackTrace();  }  **catch** (ClassNotFoundException e)  {  e.printStackTrace();  }  **catch** (Exception e)  {  e.printStackTrace();  }  **finally**  {  /\*  \* finally block used to close resources  \*/  **try**  {  **if** (callableStatement != **null**)  {  callableStatement.close();  }  }  **catch** (SQLException sqlException)  {  sqlException.printStackTrace();  }  **try**  {  **if** (connection != **null**)  {  connection.close();  }  }  **catch** (SQLException sqlException)  {  sqlException.printStackTrace();  }  }  }  } |

### c) Output from Java class:

|  |
| --- |
| When I entered the ReginId 44, I got the result:  A screenshot of a cell phone  Description generated with very high confidence  Here we can see more results from java class:  A screenshot of a cell phone  Description generated with very high confidence |

## 4.3 JDBC - CallableStatement Cursor(Oracle)

### Stored procedure:

|  |
| --- |
| create or replace  PROCEDURE GET\_ALL\_COUNTRIES  (  country\_cursor OUT SYS\_REFCURSOR  ) AS  BEGIN  OPEN country\_cursor FOR  SELECT \* FROM COUNTRIES ;  END GET\_ALL\_COUNTRIES;  /  A screenshot of a cell phone  Description generated with very high confidence |

### b) JDBC code in Eclipse to get all data from table via stored procedure:

|  |
| --- |
| **import** java.sql.CallableStatement;  **import** java.sql.Connection;  **import** java.sql.DriverManager;  **import** java.sql.ResultSet;  **import** java.sql.SQLException;  **import** oracle.jdbc.OracleTypes;  **public** **class** JDBCExample2  {  // JDBC driver name and database URL  **static** **final** String ***JDBC\_DRIVER*** = "oracle.jdbc.driver.OracleDriver";  **static** **final** String ***DB\_URL*** = "jdbc:oracle:thin:@localhost:1521:orcl";  // Database credentials  **static** **final** String ***USERNAME*** = "system";  **static** **final** String ***PASSWORD*** = "123456789";  **public** **static** **void** main(String[] args)  {  JDBCExample2 jdbcCallableStatementDemo = **new** JDBCExample2();  jdbcCallableStatementDemo.getAllCountryInformation();  }  **private** **void** getAllCountryInformation()  {  Connection connection = **null**;  CallableStatement callableStatement = **null**;  **try**  {  /\*  \* Register the JDBC driver in DriverManager  \*/  Class.*forName*(***JDBC\_DRIVER***);  /\*  \* Establish connection to the Database using DriverManager  \*/  connection = DriverManager  .*getConnection*(***DB\_URL***, ***USERNAME***, ***PASSWORD***);  String plSql = "{call GET\_ALL\_COUNTRIES(?)}";  callableStatement = connection.prepareCall(plSql);    callableStatement.registerOutParameter(1, OracleTypes.***CURSOR***);  /\*  \* Use execute method to run the stored procedure.  \*/  callableStatement.executeQuery();  ResultSet rs=(ResultSet)callableStatement.getObject(1);    **while**(rs.next())  {  System.***out***.print("COUNTRY\_ID : "+rs.getString(1));  System.***out***.print(", COUNTRY\_NAME : "+rs.getString(2));  System.***out***.println(", REGION\_ID : "+rs.getString(3));  }  }  **catch** (SQLException se)  {  se.printStackTrace();  }  **catch** (ClassNotFoundException e)  {  e.printStackTrace();  }  **catch** (Exception e)  {  e.printStackTrace();  }  **finally**  {  /\*  \* finally block used to close resources  \*/  **try**  {  **if** (callableStatement != **null**)  {  callableStatement.close();  }  }  **catch** (SQLException sqlException)  {  sqlException.printStackTrace();  }  **try**  {  **if** (connection != **null**)  {  connection.close();  }  }  **catch** (SQLException sqlException)  {  sqlException.printStackTrace();  }  }  }  } |

### c) Output from Java class in Eclipse:

|  |
| --- |
| A screenshot of a cell phone  Description generated with very high confidence |

# Inserting Data from Java class (JDBC):

## Creation of Student table:

|  |
| --- |
| CREATE TABLE STUDENTS(  ID NUMBER,  NAME VARCHAR2(20),  COUNTRY VARCHAR(20),  MARK NUMBER,  DEGREE VARCHAR2(20)  );  A screenshot of a cell phone  Description generated with very high confidence |

## JDBC code:

|  |
| --- |
| **import** java.sql.\*;  **public** **class** InsertExample {  **public** **static** **void** main(String[] args) **throws** SQLException {    Connection conn = **null**;  **try** {  **int** id[] = {1, 2, 3, 4};  String name[] = {"Dilmurod", "Sharme Kumar", "king ku chu", "Alp Arslong"};  String country\_name[] = {"Uzbekistan","India","Japan", "Turkey"};  **int** MARK[] = {10, 9, 7, 8};  String degree[] = {"Bachelor", "Bachelor", "Master", "Bachelor"};    DriverManager.*registerDriver*(**new** oracle.jdbc.driver.OracleDriver());    conn = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:orcl",  "system", "d7270895");  String sql = "INSERT INTO STUDENTS VALUES (?,?,?,?,?)";  PreparedStatement pstmt = conn.prepareStatement(sql);  **for** (**int** i=0; i<4; i++)  {    pstmt.setInt(1, id[i]);  pstmt.setString(2, name[i]);  pstmt.setString(3, country\_name[i]);  pstmt.setInt(4, MARK[i]);  pstmt.setString(5, degree[i]);  pstmt.executeUpdate();    System.***out***.println("Data inserted!!!");    }  pstmt.close();  conn.close();  } **catch** (SQLException e) {    System.***err***.println(e.getMessage());  conn.close();  }  }  }  A screenshot of a cell phone  Description generated with very high confidence |

## Output of data with select to see the data

|  |
| --- |
| **QUERY\_1#:**  SELECT \* FROM STUDENTS;  A screenshot of a cell phone  Description generated with very high confidence |

# Select statement from Java method:

## Java code:

|  |
| --- |
| **import** java.sql.\*;  **import** java.sql.DriverManager;  **import** java.sql.PreparedStatement;  **import** java.sql.SQLException;  **public** **class** RetrieveData {  **public** **static** **void** main(String[] args) **throws** SQLException {    Connection conn = **null**;  Statement stmt = **null**;  **try** {    DriverManager.*registerDriver*(**new** oracle.jdbc.driver.OracleDriver());    conn = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521:orcl",  "system", "d7270895");  String sql = "SELECT \* FROM STUDENTS";  stmt = conn.createStatement();  ResultSet rs = stmt.executeQuery(sql);    **while** (rs.next()) {  **int** id = rs.getInt("ID");  String name = rs.getString("NAME");  String country = rs.getString("COUNTRY");  **int** mark = rs.getInt("MARK");  String degree = rs.getString("DEGREE");  System.***out***.println(id + "\t" + name + "\t" + country + "\t\t" + mark +  "\t" + degree);    }    conn.close();  } **catch** (SQLException e) {    System.***err***.println(e.getMessage());  conn.close();  }  }  } |

## Output:

|  |
| --- |
| A screenshot of a cell phone  Description generated with very high confidence |

# Conclusion

In this practical work, I have learned Java Database Connectivity and how to connect JDBC to oracle by using two ways. I had a lot of error and those errors stopped me a lot. I have many errors like **ORA-12505, TNS:listener does not currently know of SID given in connect des** and **java.sql.SQLException: ORA-01017: invalid username/password; logon denied** When I wanted to connect to Oracle driver, I could not connect at the beginning and I have forgotten my password and SID, then I have fixed these issues.

Finally, I need to say that I have gained a lot of knowledge from this practical work and I hope I will use these skills in the future.