KNU CSE
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

LAB 6: DATABASE PROGRAMMING

Some from Chapter 10

- Approaches to Database Programming
- Database Programming with Function Calls: JDBC

Database Programming

- Database Applications
  - Host language
    - **JAVA**, C/C++/C#, COBOL, some other programming language
  - Data sublanguage
    - SQL
- Techniques and Issues
  - Interactive interface: SQL commands typed directly into a monitor
    - E.g., `sqlplus` (or `SQL Developer`), `mysql shell`, `psql`, `Db2`, …
  - Execute a file of commands: `@`<*filename*>`.sql` (in Oracle)
  - **Application programs** or **database applications**
    - Having access to databases via DBMS from a user program or a web interface

Approaches to Database Programming

1) Embedding database commands in a general-purpose programming language (common)

- Database statements are *embedded* into the host programing language.
  - Such statements identified by a special prefix, or **`EXEC SQL`**.
- Precompiler or preprocessor scans the source program code.
  - Identify database statements and extract them for processing by the DBMS
- This technique is generally referred to as **embedded SQL**.

Approaches to Database Programming (Cont'd)

1) Embedded SQL Example in C

```
int loop ;
EXEC SQL BEGIN DECLARE SECTION ;
varchar dname [16], fname [16], lname [16], address [31] ;
char ssn [10], bdate [11], sex [2], minit [2] ;
float salary, raise ;
int dno, dnumber ;
int SQLCODE ; char SQLSTATE [6] ;
EXEC SQL END DECLARE SECTION ;
loop = 1 ;
while (loop) {
  prompt("Enter a Social Security Number: ", ssn) ;
  EXEC SQL
    SELECT Fname, Minit, Lname, Address, Salary
    INTO :fname, :minit, :lname, :address, :salary
    FROM EMPLOYEE WHERE Ssn = :ssn ;
  if (SQLCODE = = 0) printf(fname, minit, lname, address, salary)
    else printf("Social Security Number does not exist: ", ssn) ;
  prompt("More Social Security Numbers (enter 1 for Yes, 0 for No): ", loop) ;
  }
```

Approaches to Database Programming (Cont'd)

2) Designing a brand-new language (less common)

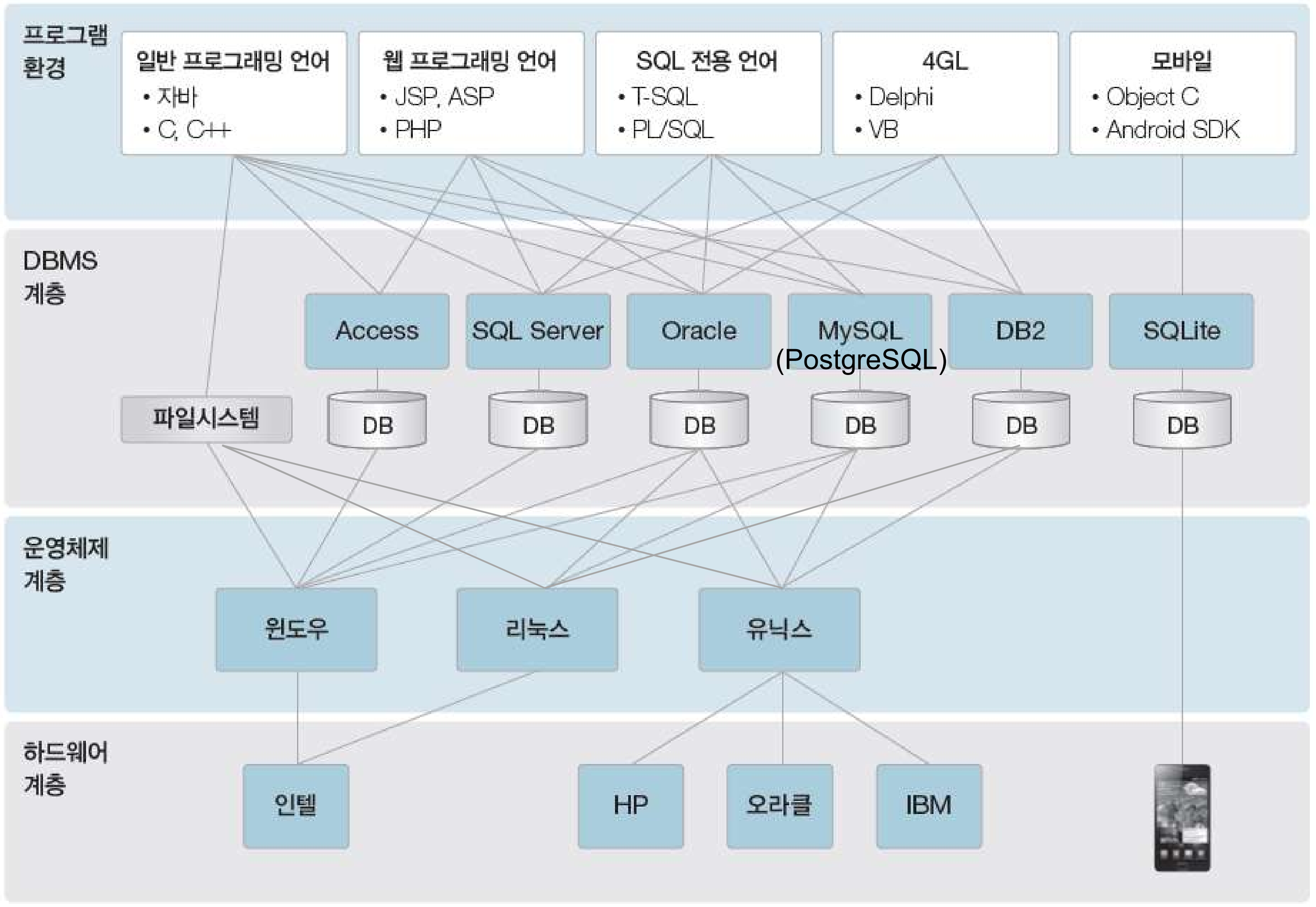
- "Database(-exclusive)" programming language (DBPL) is designed from scratch to be compatible with the database model and query language
  - More appropriate for applications that have intensive DB applications
- Additional programming structures such as loops and conditional statements are "added" to the database programming language.
  - Why? *To convert it into a full-fledged (or complete) programming language*
- Classic DBPL example: Oracle's PL/SQL, SQL Server's T-SQL
  - Hopefully, PL/SQL with PSM will be experienced in your lab next week.

Approaches to Database Programming (Cont'd)

3) Using a **library** of database functions (common)

- Library of functions available to the host programming language (e.g., JAVA or C(++)): called ***A**pplication **P**rogramming **I**nterface* (API)
- The library consists of a variety of functions accessing a DBMS
  - **ODBC** (Open DataBase Connectivity): Version 4 as of June 2016
    - For C language
  - **JDBC** (Java DataBase Connectivity): Version 4.3 as of September 2017
    - Included in Java SE 9
    - For JAVA language; impedance mismatch greatly reduced by using a Java-compliant object database

[Appendix] DBMS Platform & DB Programming Type

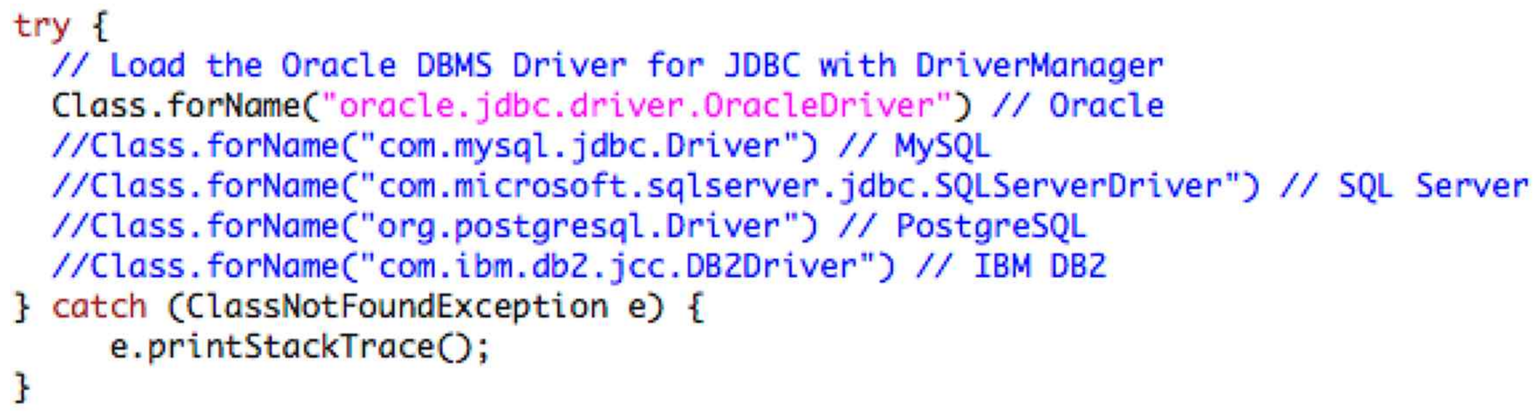


JDBC PROGRAMMING

Some slides contributed by 튜터s 이태현, 허철훈

JDBC

- Designed to access a variety of DBMSs (Oracle, DB2, SQL Server, MySQL, PostgreSQL, JavaDB, Sqlite, SAP HANA…).
  - Main advantage: A single (Java) program can connect to multiple databases and do its task on top of them.
  - DBMS-independent, but you first need to explicitly load an appropriate driver associated with a chosen DBMS



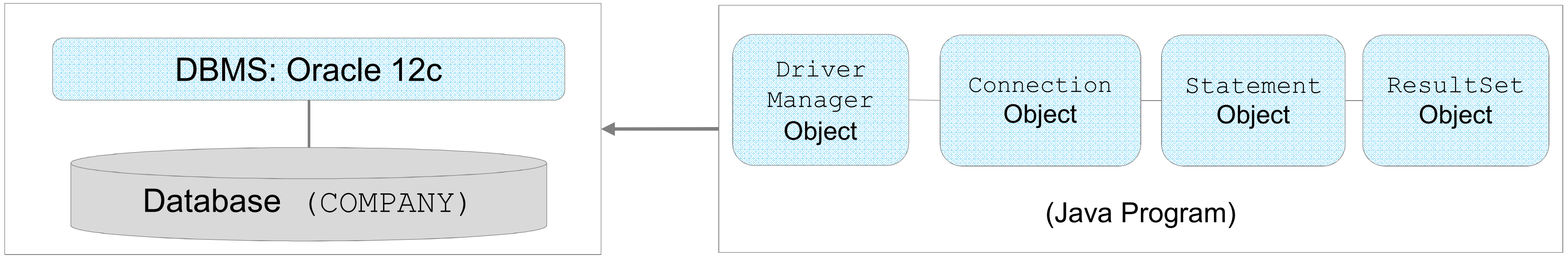
```
try {
  // Load the Oracle DBMS Driver for JDBC with DriverManager
  Class.forName("oracle.jdbc.driver.OracleDriver") // Oracle
  //Class.forName("com.mysql.jdbc.Driver") // MySQL
  //Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver") // SQL Server
  //Class.forName("org.postgresql.Driver") // PostgreSQL
  //Class.forName("com.ibm.db2.jcc.DB2Driver") // IBM DB2
} catch (ClassNotFoundException e) {
     e.printStackTrace();
}
```

JDBC Programming Environment

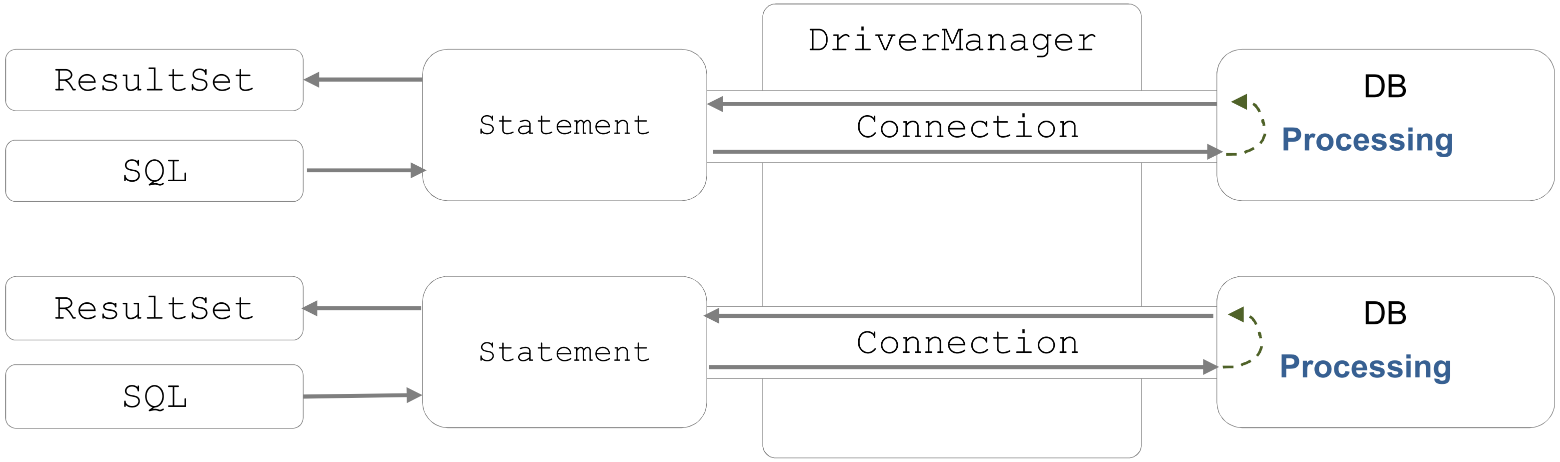
Item	System/Program
DBMS	Oracle DBMS 12c
Javac	JDK 버전 9 (or 8)
JDBC Driver	Oracle JDBC Library: `ojdbc7.jar` (Should fine with `ojdbc6.jar`.)

(In the subsequent slides, we'll set up the Oracle JDBC Driver.)

[Appendix] Relationship among Java Objects for Database Connection (in JDBC)



[Java Objects for Database Access]

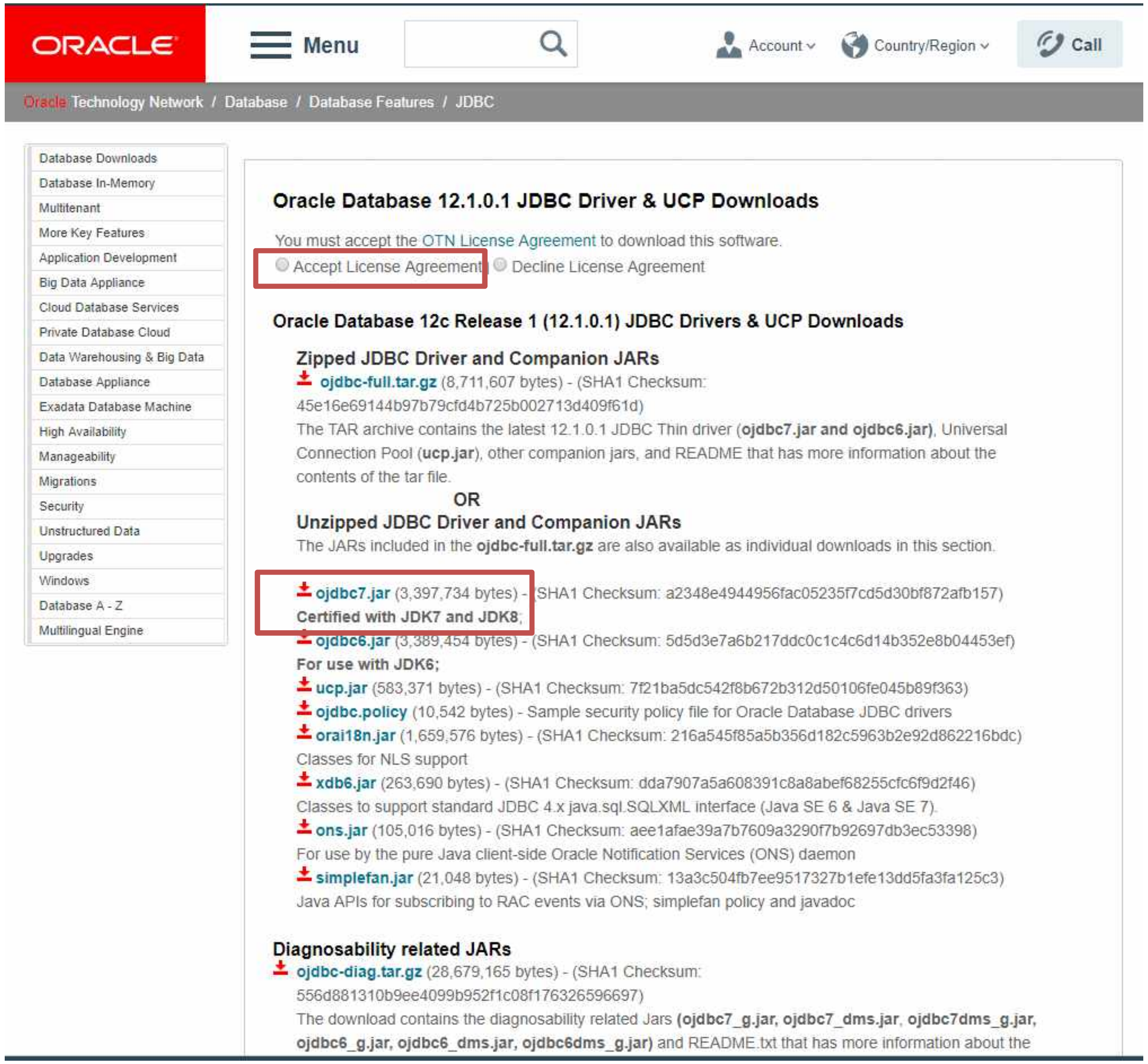


[A Calling Order of Java Objects]

[Appendix] Major Packages Related to JDBC

Class	Class/Interface	Major Methods	Description
java.lang	Class	Class forName(<클래스이름>)	<클래스이름>의 JDBC 드라이버를 로딩
java.sql	DriverManager	Connection getConnection (url, user, password)	데이터베이스 Connection 객체를 생성
	Connection	Statement createStatement()	SQL 문을 실행하는 Statement 객체를 생성
		void close()	Connection 객체 연결을 종료
	Statement (inherited by PreparedStatement, CallableStatement)	ResultSet executeQuery (String sql)	SQL 문을 실행해서 ResultSet 객체를 생성
		ResultSet executeUpdate (String sql)	INSERT/DELETE/UPDATE 문을 실행해서 ResultSet 객체를 생성
		void close()	Statement 객체 사용을 종료
	ResultSet	boolean first()	결과 테이블에서 커서가 처음 투플을 가리킴
		boolean next()	결과 테이블에서 커서가 다음 투플을 가리킴
		int getInt(<int>)	<int>가 가리키는 열 값을 정수로 반환
		String getString(<int>)	<int>가 가리키는 열 값을 문자열로 반환

https://www.oracle.com/technetwork/database/features/jdbc/jdbc-drivers-12c-download-1958347.html



Oracle Database 12.1.0.1 JDBC Driver & UCP Downloads

You must accept the OTN License Agreement to download this software.

Accept License Agreement   Decline License Agreement

Oracle Database 12c Release 1 (12.1.0.1) JDBC Drivers & UCP Downloads

Zipped JDBC Driver and Companion JARs

ojdbc-full.tar.gz (8,711,607 bytes) - (SHA1 Checksum: 45e16e69144b97b79cfd4b725b002713d409f61d)
The TAR archive contains the latest 12.1.0.1 JDBC Thin driver (**ojdbc7.jar and ojdbc6.jar**), Universal Connection Pool (**ucp.jar**), other companion jars, and README that has more information about the contents of the tar file.

OR

Unzipped JDBC Driver and Companion JARs

The JARs included in the **ojdbc-full.tar.gz** are also available as individual downloads in this section.

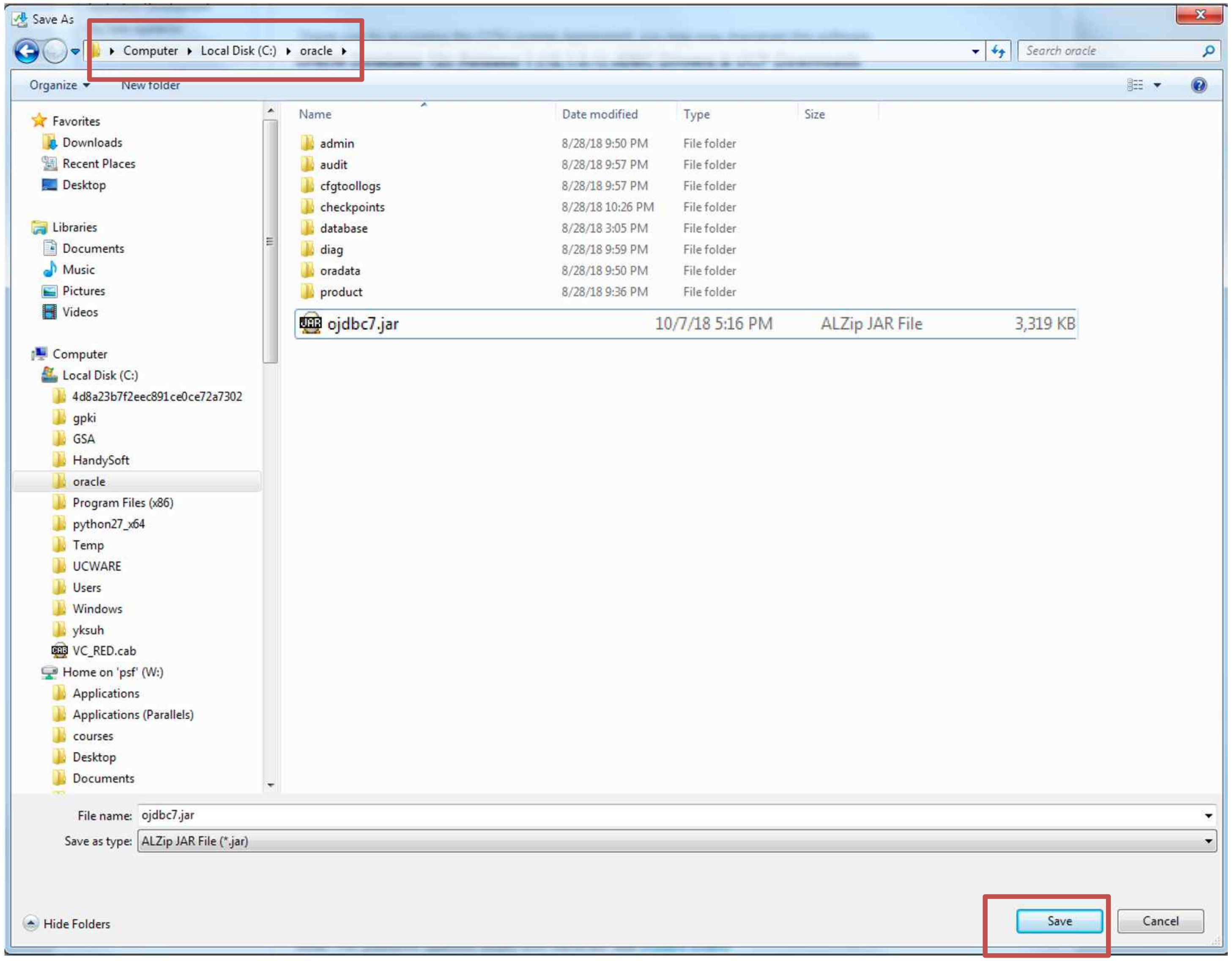
ojdbc7.jar (3,397,734 bytes) - (SHA1 Checksum: a2348e4944956fac05235f7cd5d30bf872afb157)
Certified with JDK7 and JDK8;
ojdbc6.jar (3,389,454 bytes) - (SHA1 Checksum: 5d5d3e7a6b217ddc0c1c4c6d14b352e8b04453ef)
For use with JDK6;
ucp.jar (583,371 bytes) - (SHA1 Checksum: 7f21ba5dc542f8b672b312d50106fe045b89f363)
ojdbc.policy (10,542 bytes) - Sample security policy file for Oracle Database JDBC drivers
orai18n.jar (1,659,576 bytes) - (SHA1 Checksum: 216a545f85a5b356d182c5963b2e92d862216bdc)
Classes for NLS support
xdb6.jar (263,690 bytes) - (SHA1 Checksum: dda7907a5a608391c8a8abef68255cfc6f9d2f46)
Classes to support standard JDBC 4.x java.sql.SQLXML interface (Java SE 6 & Java SE 7).
ons.jar (105,016 bytes) - (SHA1 Checksum: aee1afae39a7b7609a3290f7b92697db3ec53398)
For use by the pure Java client-side Oracle Notification Services (ONS) daemon
simplefan.jar (21,048 bytes) - (SHA1 Checksum: 13a3c504fb7ee9517327b1efe13dd5fa3fa125c3)
Java APIs for subscribing to RAC events via ONS; simplefan policy and javadoc

Diagnosability related JARs

ojdbc-diag.tar.gz (28,679,165 bytes) - (SHA1 Checksum: 556d881310b9ee4099b952f1c08f176326596697)
The download contains the diagnosability related Jars (**ojdbc7_g.jar, ojdbc7_dms.jar, ojdbc7dms_g.jar, ojdbc6_g.jar, ojdbc6_dms.jar, ojdbc6dms_g.jar**) and README.txt that has more information about the

Download the Oracle JDBC Driver

Installing the Oracle JDBC Driver

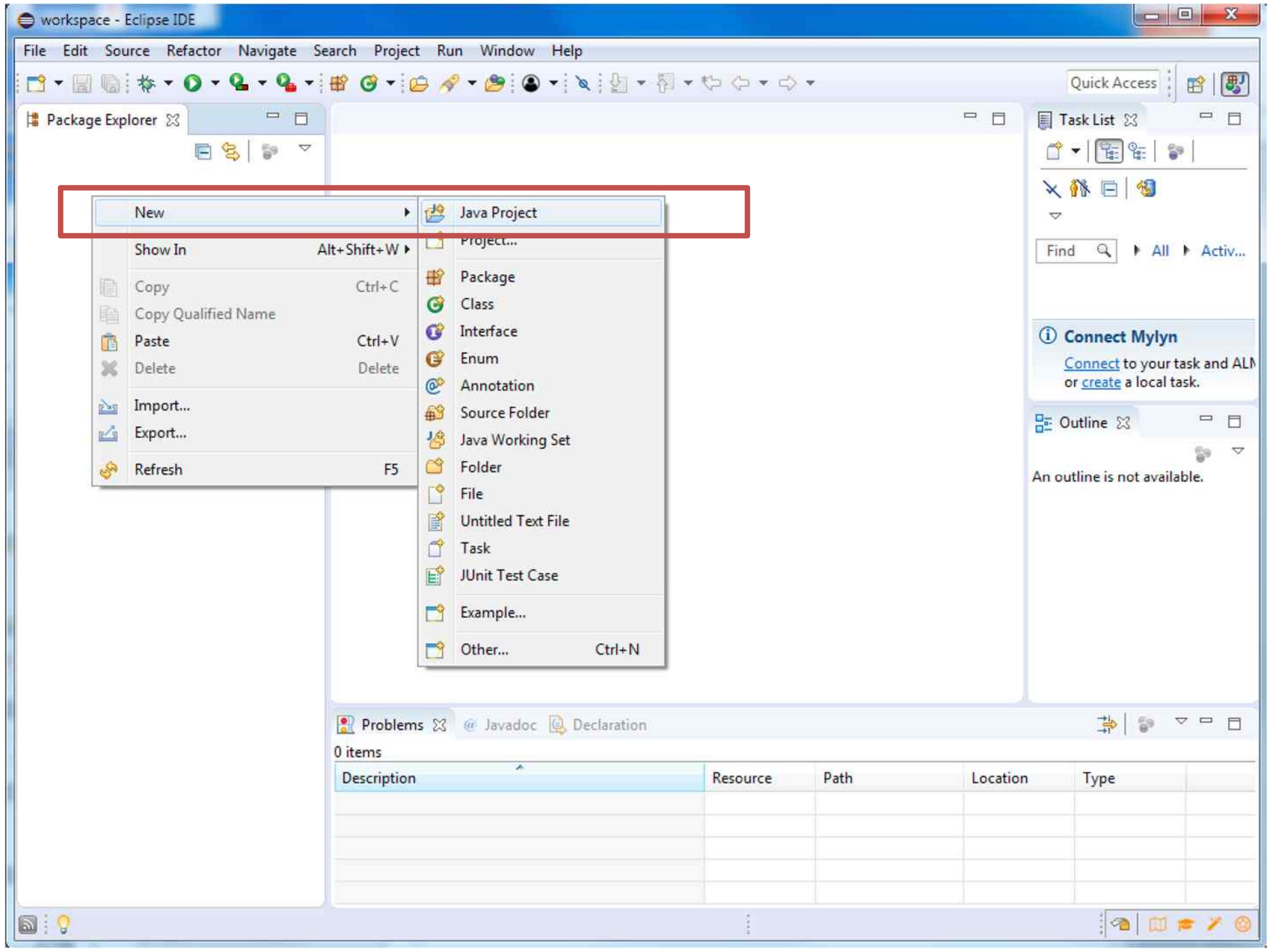


Name	Date modified	Type	Size
admin	8/28/18 9:50 PM	File folder	
audit	8/28/18 9:57 PM	File folder	
cfgtoollogs	8/28/18 9:57 PM	File folder	
checkpoints	8/28/18 10:26 PM	File folder	
database	8/28/18 3:05 PM	File folder	
diag	8/28/18 9:59 PM	File folder	
oradata	8/28/18 9:50 PM	File folder	
product	8/28/18 9:36 PM	File folder	
ojdbc7.jar	10/7/18 5:16 PM	ALZip JAR File	3,319 KB

File name: ojdbc7.jar

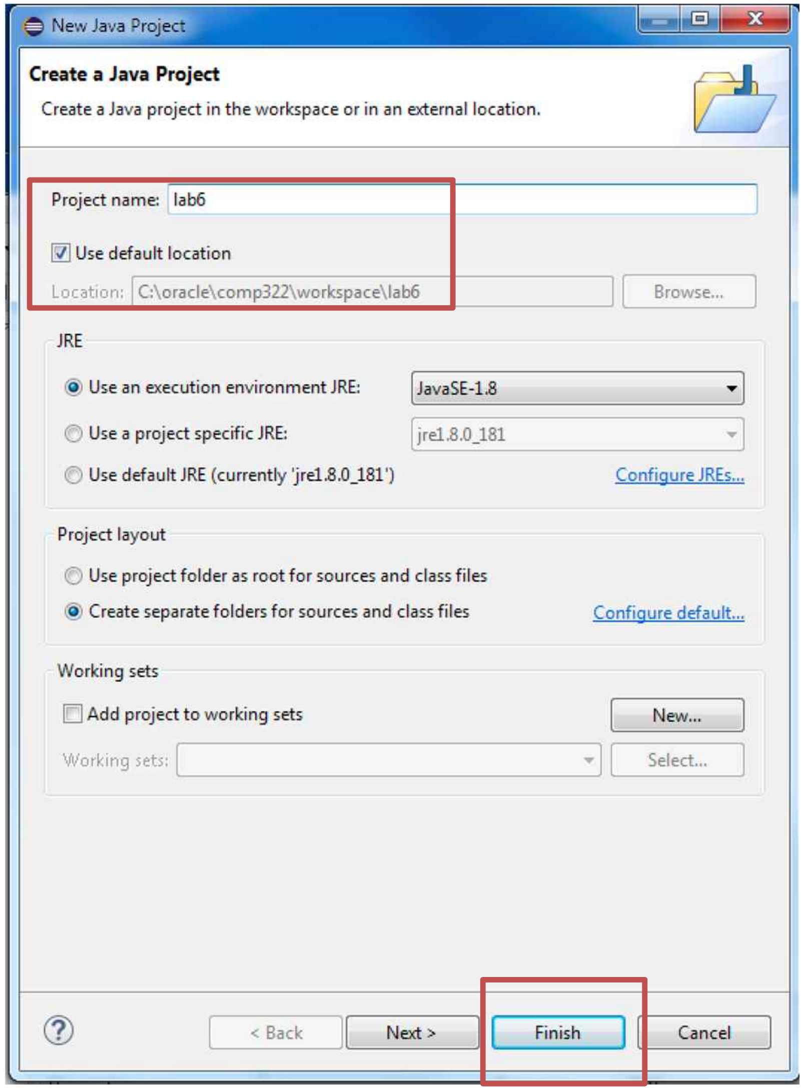
Save as type: ALZip JAR File (*.jar)

Creating a New Project in Eclipse



Now let's set up the driver in the following way: `Project->Properties->Libraries ->Add External Jars->`Select '`ojdbc7.jar`'.

Creating a New Project in Eclipse (Cont'd)


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: lab6

[✓] Use default location

Location: C:\oracle\comp322\workspace\lab6

JRE

(●) Use an execution environment JRE: JavaSE-1.8

( ) Use a project specific JRE: jre1.8.0_181

( ) Use default JRE (currently 'jre1.8.0_181')

Project layout

( ) Use project folder as root for sources and class files

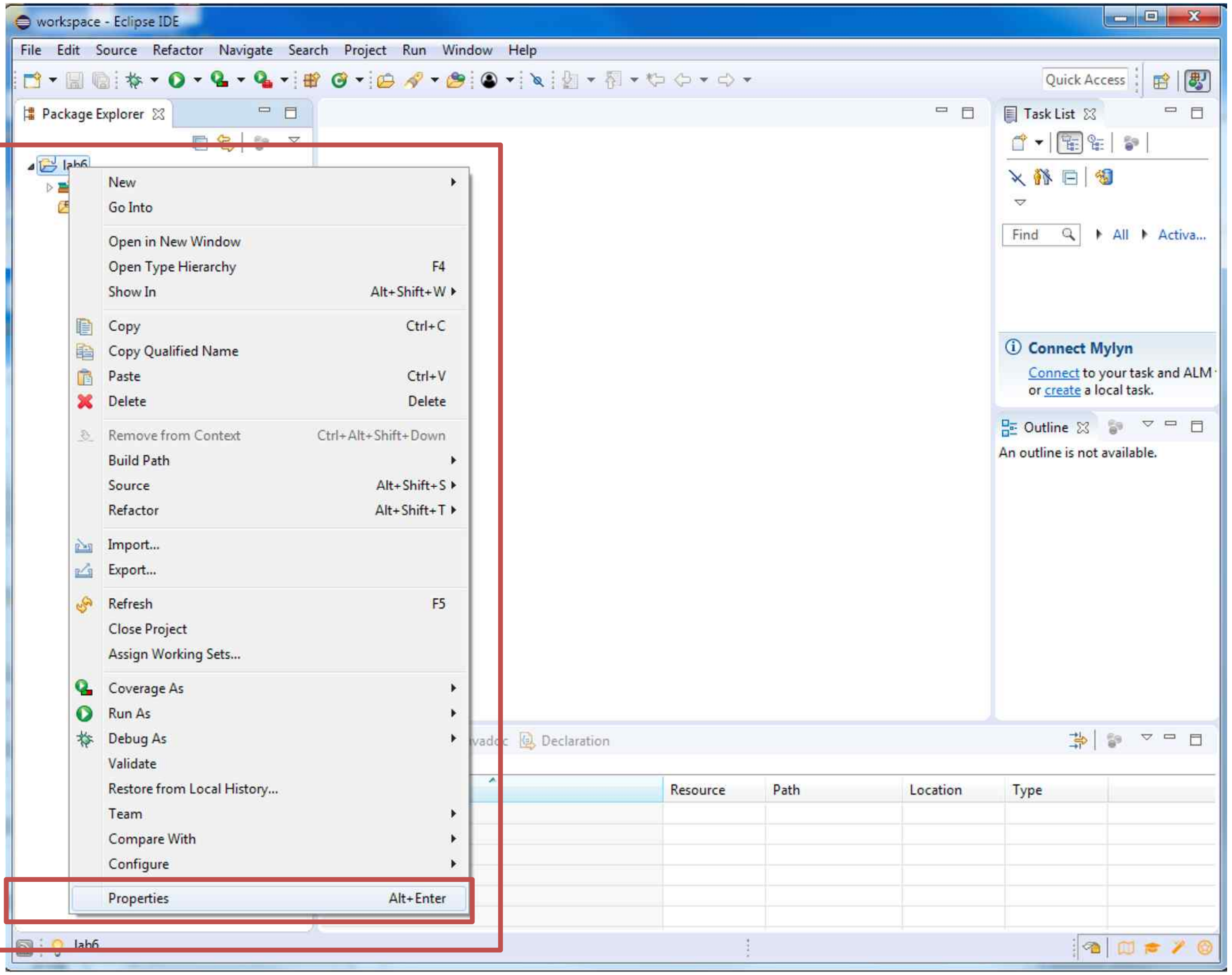
(●) Create separate folders for sources and class files

Working sets

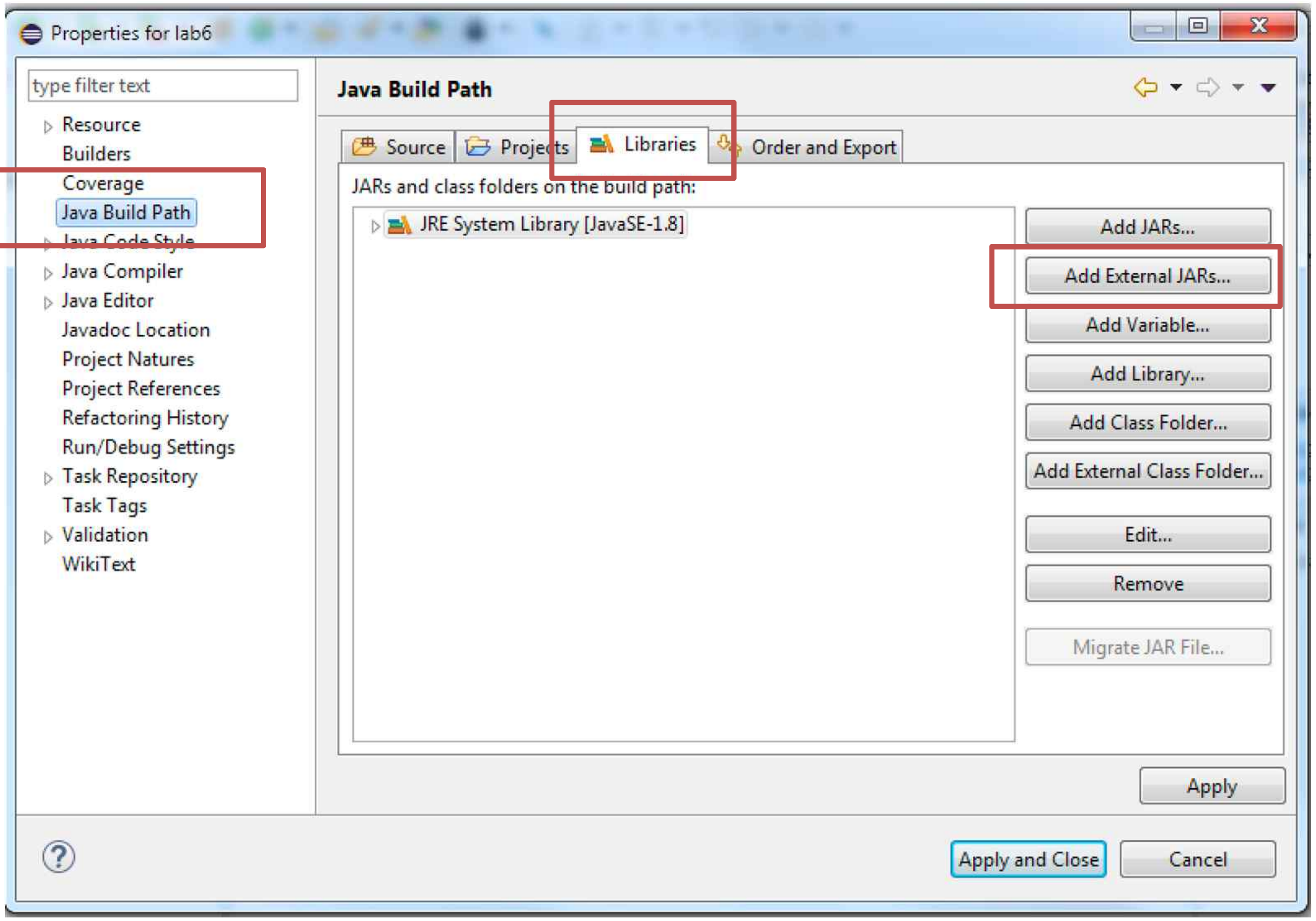
[ ] Add project to working sets

Working sets:

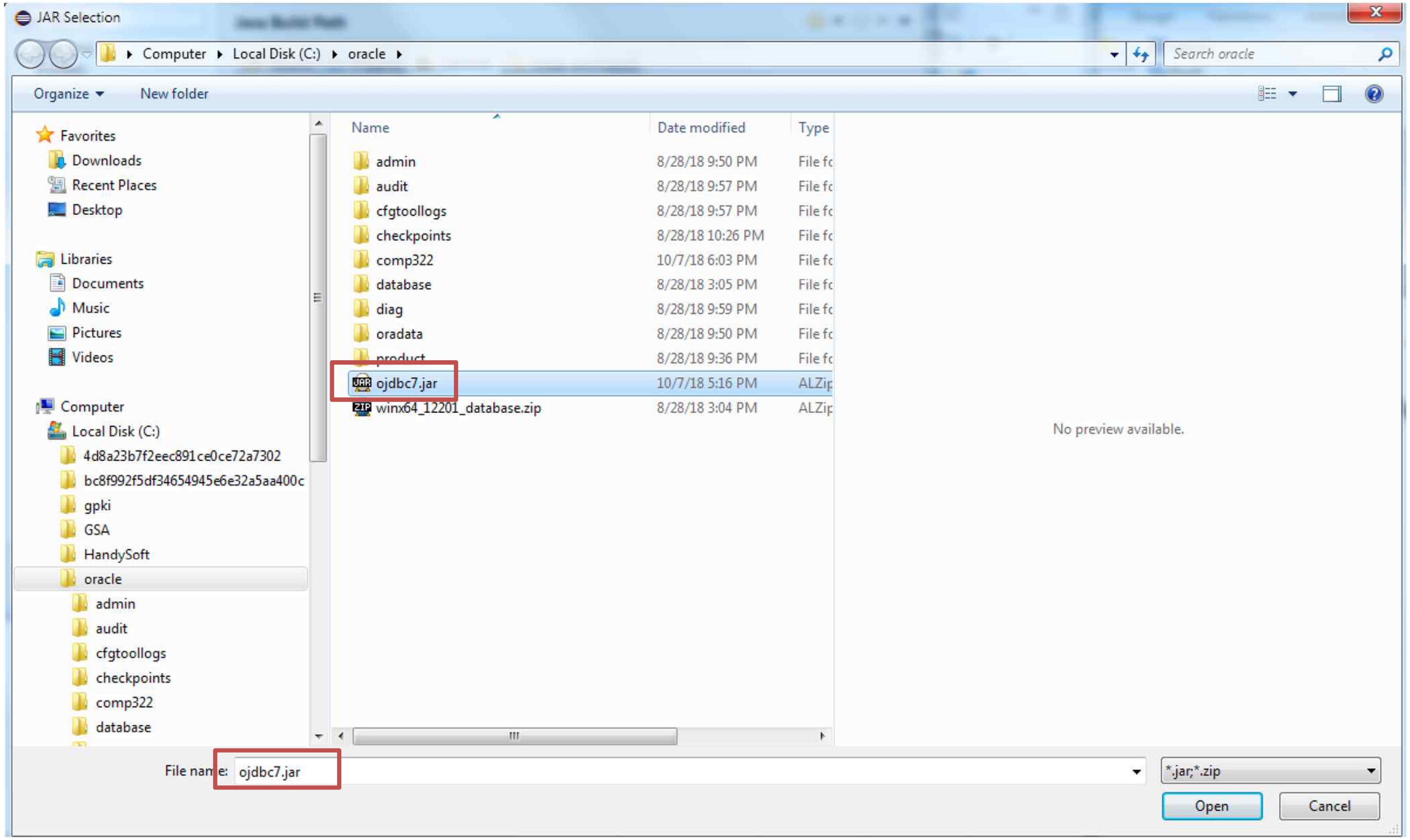
Import Oracle JDBC Driver in Eclipse



Importing Oracle JDBC Driver in Eclipse (Cont'd)



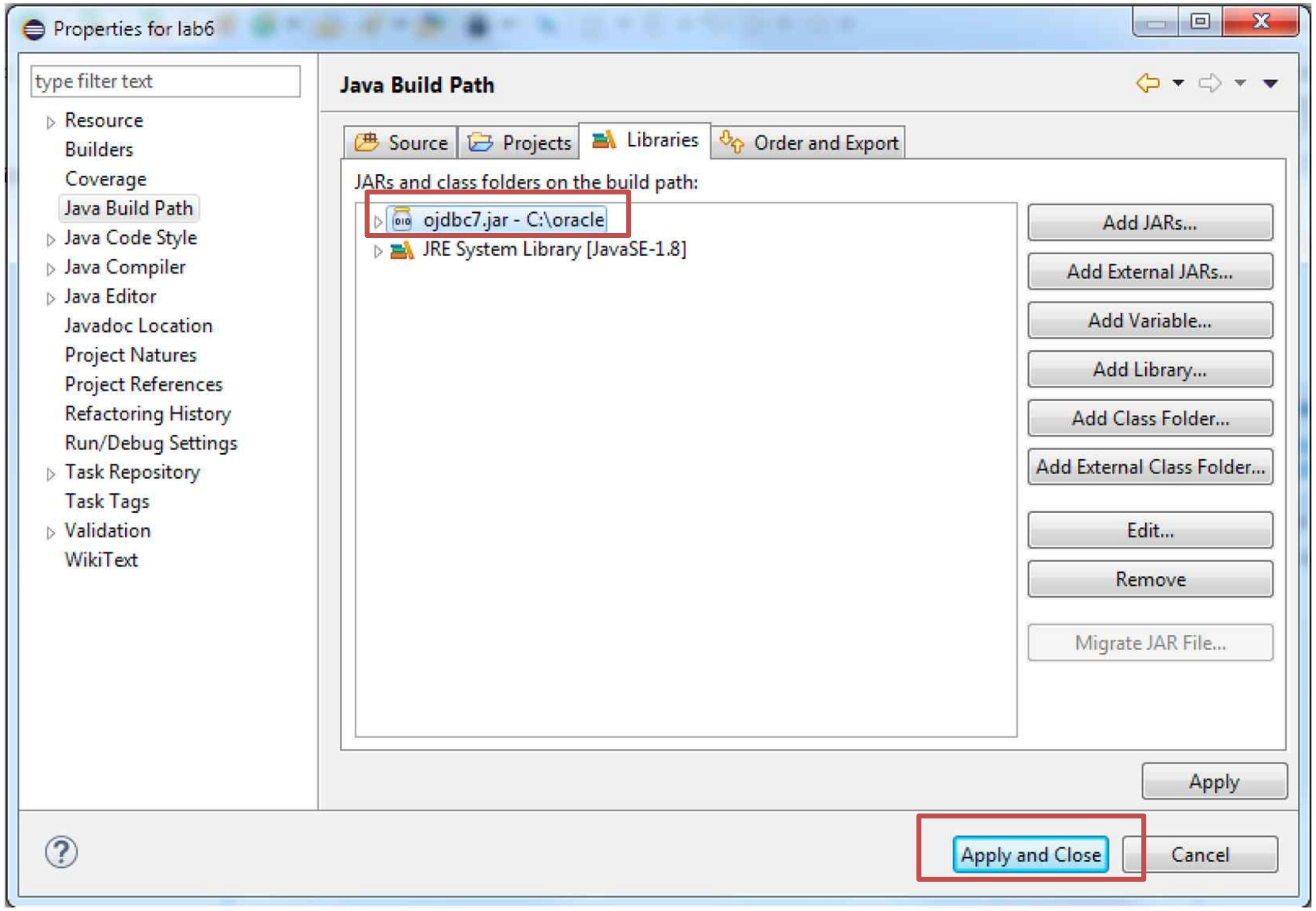
Importing Oracle JDBC Driver in Eclipse (Cont'd)



Name	Date modified	Type
admin	8/28/18 9:50 PM	File fc
audit	8/28/18 9:57 PM	File fc
cfgtoollogs	8/28/18 9:57 PM	File fc
checkpoints	8/28/18 10:26 PM	File fc
comp322	10/7/18 6:03 PM	File fc
database	8/28/18 3:05 PM	File fc
diag	8/28/18 9:59 PM	File fc
oradata	8/28/18 9:50 PM	File fc
product	8/28/18 9:36 PM	File fc
ojdbc7.jar	10/7/18 5:16 PM	ALZip
winx64_12201_database.zip	8/28/18 3:04 PM	ALZip

File name: ojdbc7.jar

Importing Oracle JDBC Driver in Eclipse (Cont'd)


Properties for lab6

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project Natures
- Project References
- Refactoring History
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

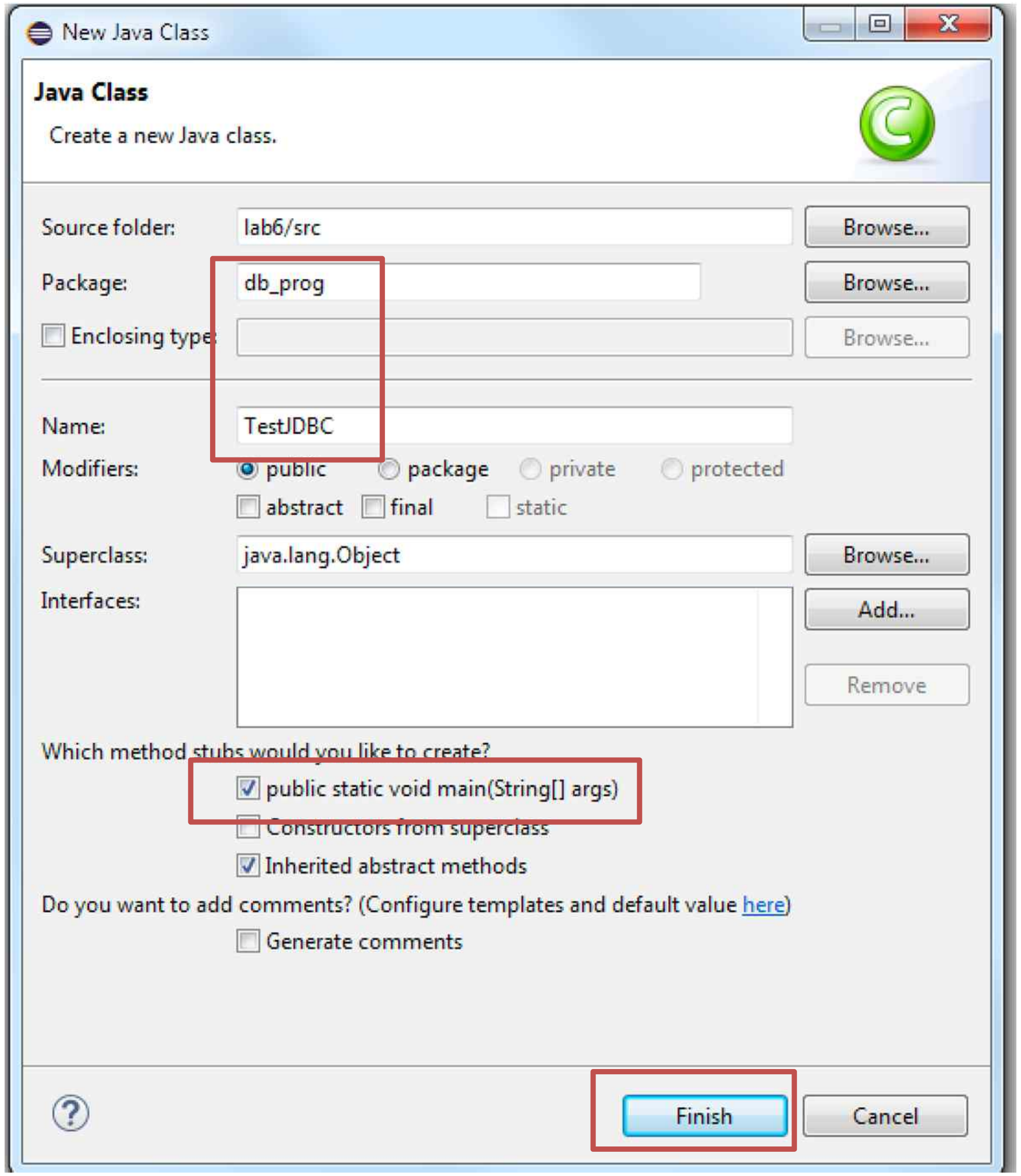
Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- ojdbc7.jar - C:\oracle
- JRE System Library [JavaSE-1.8]

Creating `TestJDBC` Java Class



New Java Class

Java Class

Create a new Java class.

Source folder: lab6/src

Package: db_prog

☐ Enclosing type:

Name: TestJDBC

Modifiers: (● public) ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

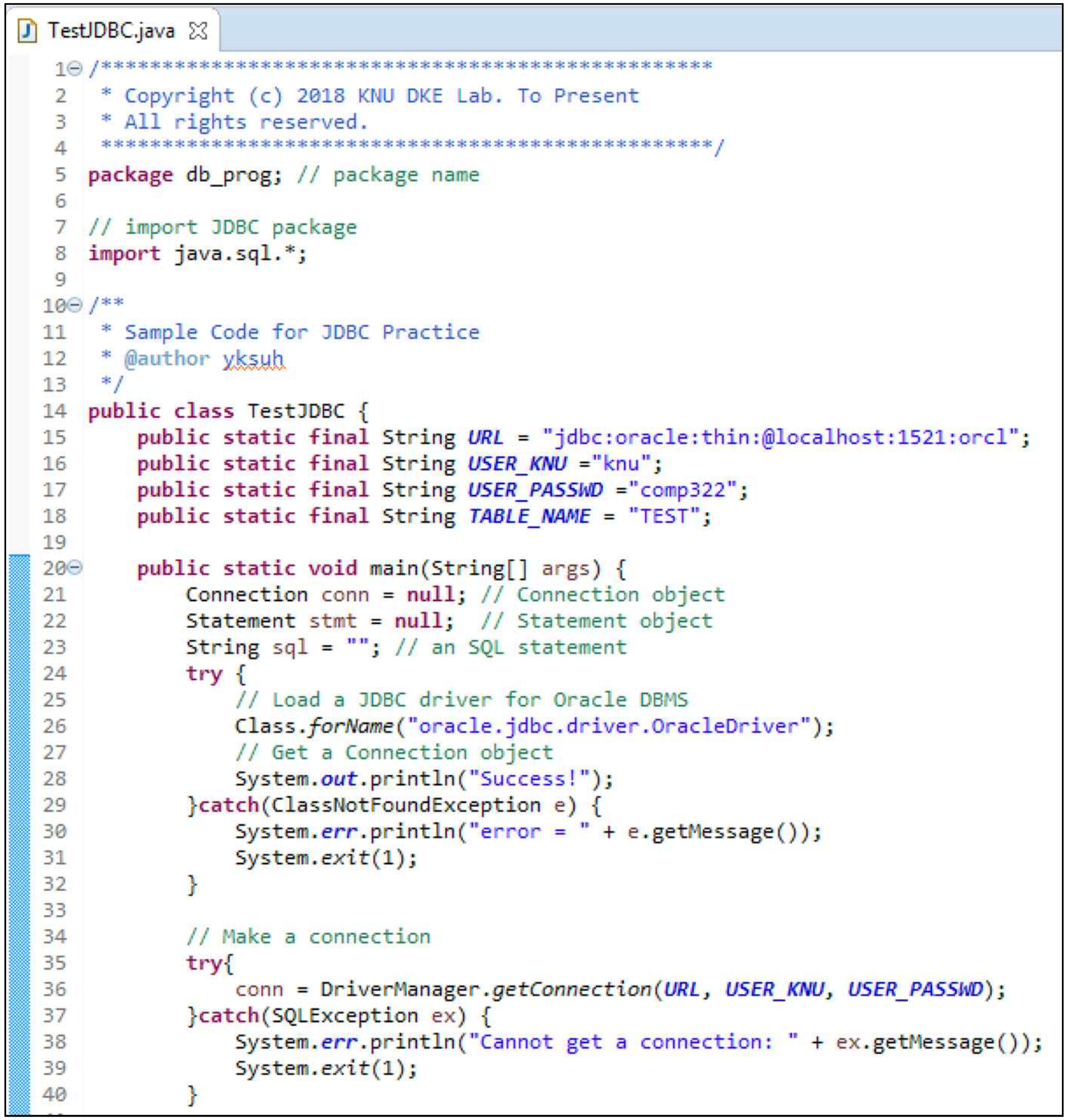
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

`TestJDBC.java`: Connection

1) Connect to Oracle DBMS.
- Load a Oracle JDBC driver.
- Get a `Connection` object.

* `TestJDBC.java`: available on LMS.

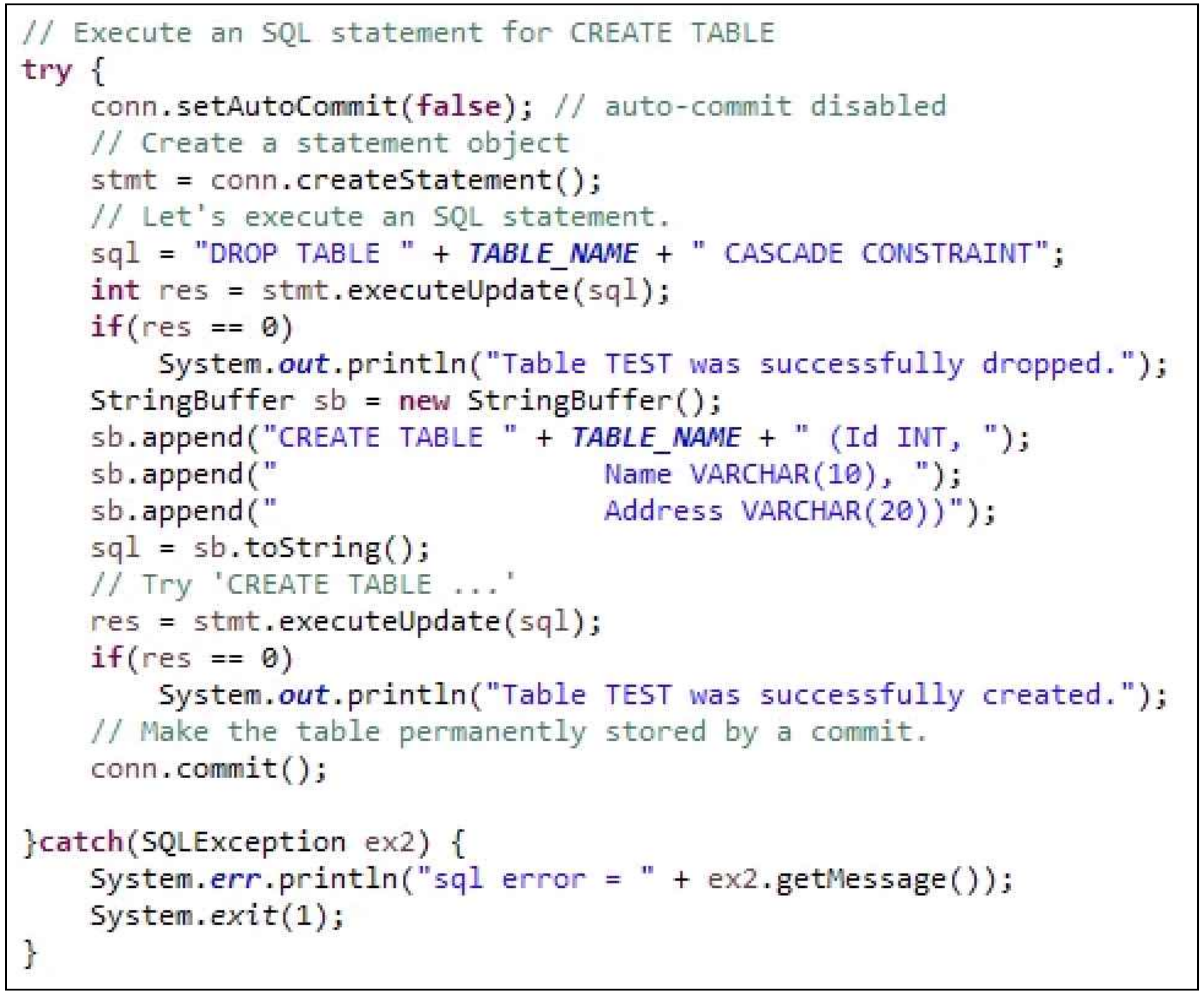

TestJDBC.java

```
/**************************************************
 * Copyright (c) 2018 KNU DKE Lab. To Present
 * All rights reserved.
 **************************************************/
package db_prog; // package name

// import JDBC package
import java.sql.*;

/**
 * Sample Code for JDBC Practice
 * @author yksuh
 */
public class TestJDBC {
    public static final String URL = "jdbc:oracle:thin:@localhost:1521:orcl";
    public static final String USER_KNU ="knu";
    public static final String USER_PASSWD ="comp322";
    public static final String TABLE_NAME = "TEST";

    public static void main(String[] args) {
        Connection conn = null; // Connection object
        Statement stmt = null;  // Statement object
        String sql = ""; // an SQL statement
        try {
            // Load a JDBC driver for Oracle DBMS
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // Get a Connection object
            System.out.println("Success!");
        }catch(ClassNotFoundException e) {
            System.err.println("error = " + e.getMessage());
            System.exit(1);
        }

        // Make a connection
        try{
            conn = DriverManager.getConnection(URL, USER_KNU, USER_PASSWD);
        }catch(SQLException ex) {
            System.err.println("Cannot get a connection: " + ex.getMessage());
            System.exit(1);
        }
```

TestJDBC.java: CREATE TABLE

2) Create a table, named `TEST`.

- Get a `Statement` object from the `Connection` object.
- Build an SQL statement: `CREATE TABLE TEST …`
- Call `executeUpdate()` with the built SQL statement.
- Do a commit.

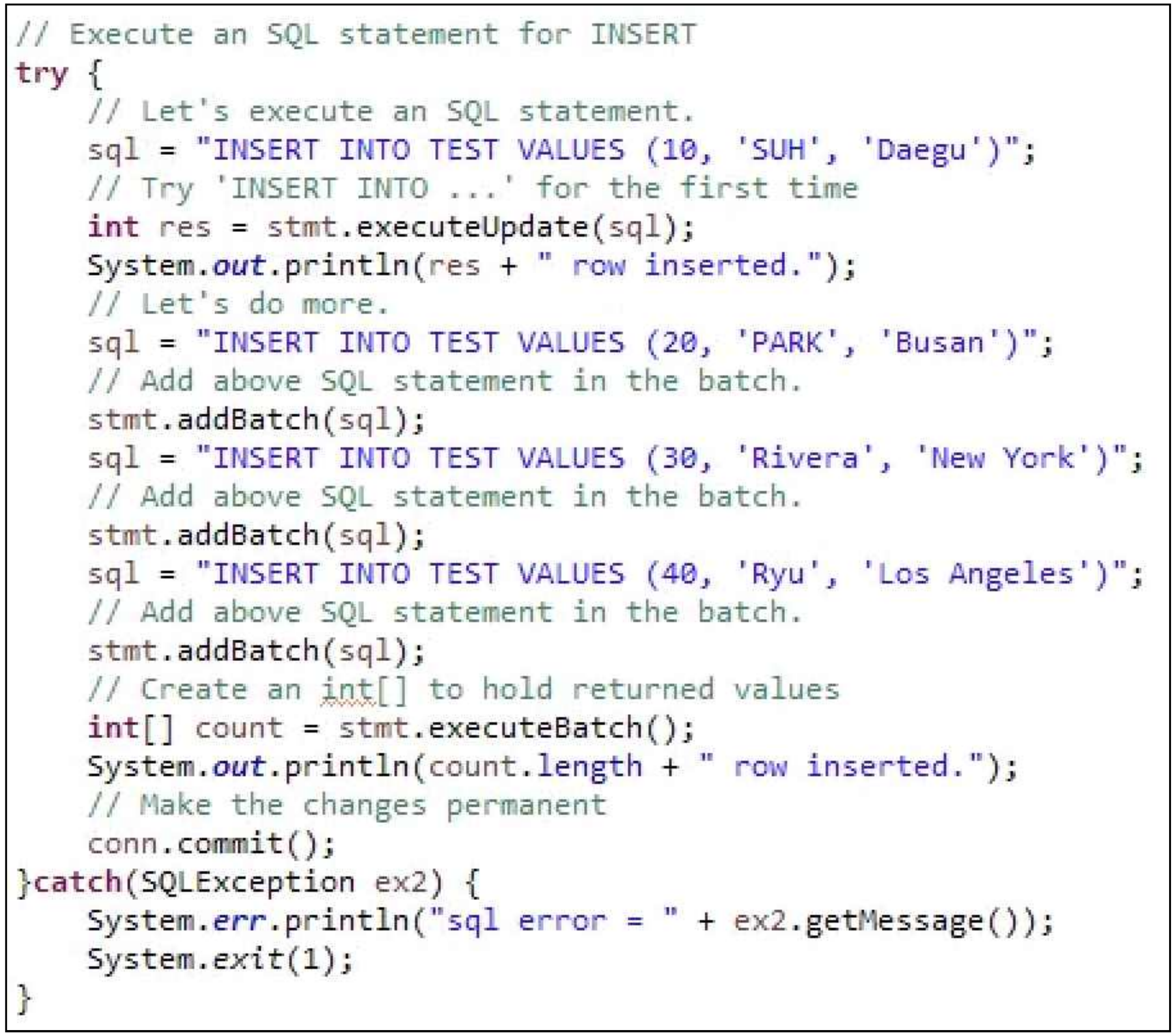


```java
// Execute an SQL statement for CREATE TABLE
try {
    conn.setAutoCommit(false); // auto-commit disabled
    // Create a statement object
    stmt = conn.createStatement();
    // Let's execute an SQL statement.
    sql = "DROP TABLE " + TABLE_NAME + " CASCADE CONSTRAINT";
    int res = stmt.executeUpdate(sql);
    if(res == 0)
        System.out.println("Table TEST was successfully dropped.");
    StringBuffer sb = new StringBuffer();
    sb.append("CREATE TABLE " + TABLE_NAME + " (Id INT, ");
    sb.append("                   Name VARCHAR(10), ");
    sb.append("                   Address VARCHAR(20))");
    sql = sb.toString();
    // Try 'CREATE TABLE ...'
    res = stmt.executeUpdate(sql);
    if(res == 0)
        System.out.println("Table TEST was successfully created.");
    // Make the table permanently stored by a commit.
    conn.commit();

}catch(SQLException ex2) {
    System.err.println("sql error = " + ex2.getMessage());
    System.exit(1);
}
```

TestJDBC.java: INSERT

3) Insert some tuples into the `TEST` table.

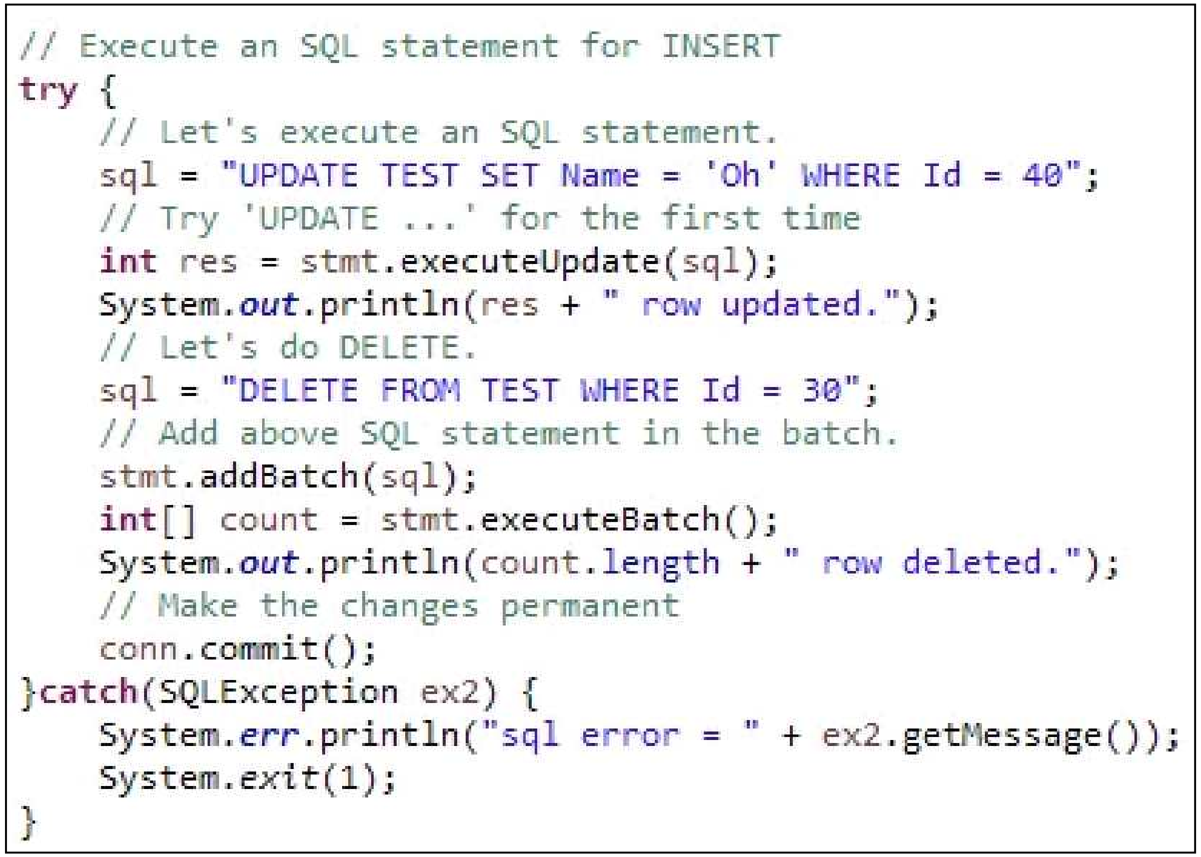
- Build an SQL statement: `INSERT INTO` …
- Use the same `Statement` object.
- Call `executeUpdate`() with the built SQL statement.
- Build other SQL statements.
- Add them in a batch via `addBatch()`.
- Invoke `executeBatch()`.
- Do a commit.



```
// Execute an SQL statement for INSERT
try {
    // Let's execute an SQL statement.
    sql = "INSERT INTO TEST VALUES (10, 'SUH', 'Daegu')";
    // Try 'INSERT INTO ...' for the first time
    int res = stmt.executeUpdate(sql);
    System.out.println(res + " row inserted.");
    // Let's do more.
    sql = "INSERT INTO TEST VALUES (20, 'PARK', 'Busan')";
    // Add above SQL statement in the batch.
    stmt.addBatch(sql);
    sql = "INSERT INTO TEST VALUES (30, 'Rivera', 'New York')";
    // Add above SQL statement in the batch.
    stmt.addBatch(sql);
    sql = "INSERT INTO TEST VALUES (40, 'Ryu', 'Los Angeles')";
    // Add above SQL statement in the batch.
    stmt.addBatch(sql);
    // Create an int[] to hold returned values
    int[] count = stmt.executeBatch();
    System.out.println(count.length + " row inserted.");
    // Make the changes permanent
    conn.commit();
}catch(SQLException ex2) {
    System.err.println("sql error = " + ex2.getMessage());
    System.exit(1);
}
```

TestJDBC.java: UPDATE/DELETE

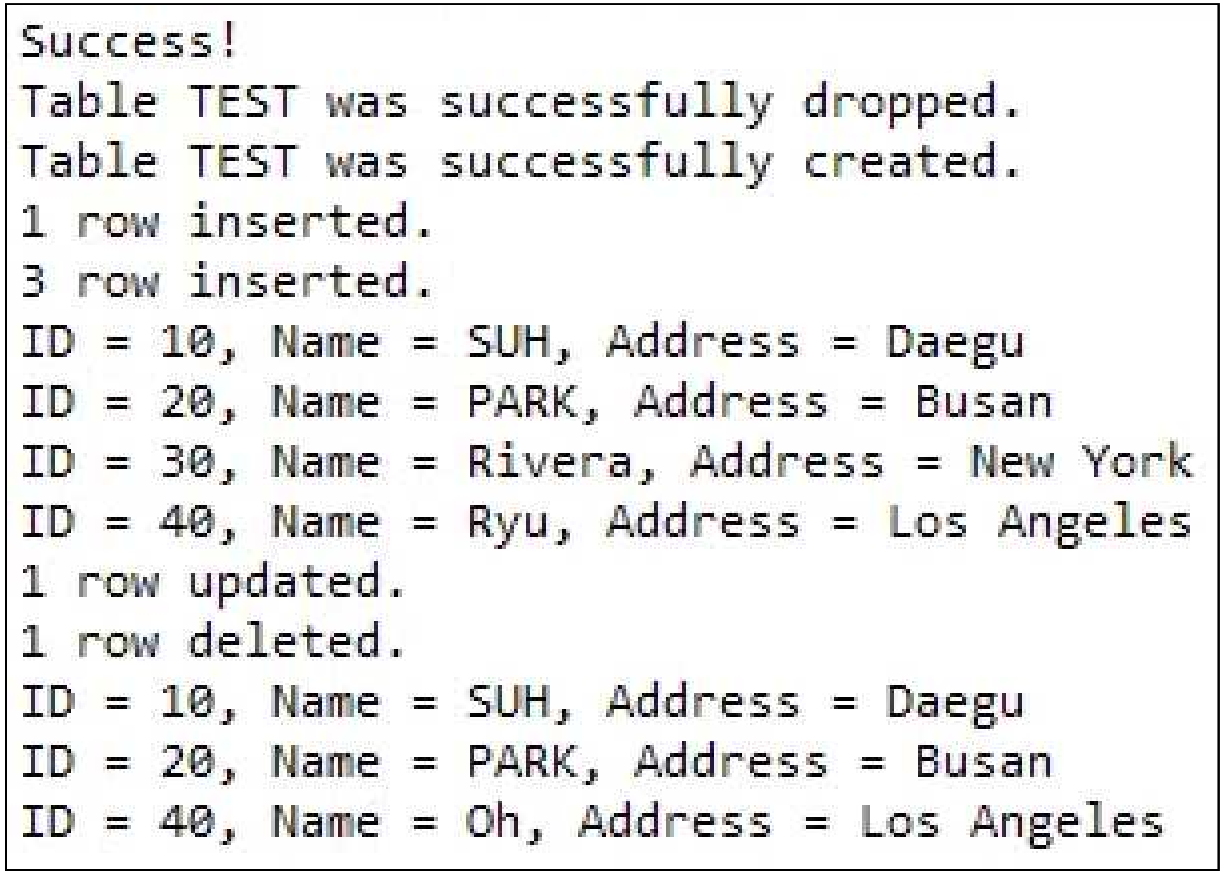
5) Update/delete some tuples in the `TEST` table.
- Build an SQL statement: `UPDATE ....`
- Use the same `Statement` object.
- Call `executeUpdate`() with the built SQL statement.
- Build an SQL statement: `DELETE ....`
- Use the same `Statement` object.
- Add the statement in a batch.
- Call `executeBatch`() with the prepared batch.
- Do a commit, optionally.



```java
// Execute an SQL statement for INSERT
try {
    // Let's execute an SQL statement.
    sql = "UPDATE TEST SET Name = 'Oh' WHERE Id = 40";
    // Try 'UPDATE ...' for the first time
    int res = stmt.executeUpdate(sql);
    System.out.println(res + " row updated.");
    // Let's do DELETE.
    sql = "DELETE FROM TEST WHERE Id = 30";
    // Add above SQL statement in the batch.
    stmt.addBatch(sql);
    int[] count = stmt.executeBatch();
    System.out.println(count.length + " row deleted.");
    // Make the changes permanent
    conn.commit();
}catch(SQLException ex2) {
    System.err.println("sql error = " + ex2.getMessage());
    System.exit(1);
}
```

Besides, I strongly recommend studying `PreparedStatement`, `CallableStatement` inherited from Statement.

`TestJDBC.java`: Check the Output



```
Success!
Table TEST was successfully dropped.
Table TEST was successfully created.
1 row inserted.
3 row inserted.
ID = 10, Name = SUH, Address = Daegu
ID = 20, Name = PARK, Address = Busan
ID = 30, Name = Rivera, Address = New York
ID = 40, Name = Ryu, Address = Los Angeles
1 row updated.
1 row deleted.
ID = 10, Name = SUH, Address = Daegu
ID = 20, Name = PARK, Address = Busan
ID = 40, Name = Oh, Address = Los Angeles
```

Lab #6: Build `COMPANY` via JDBC

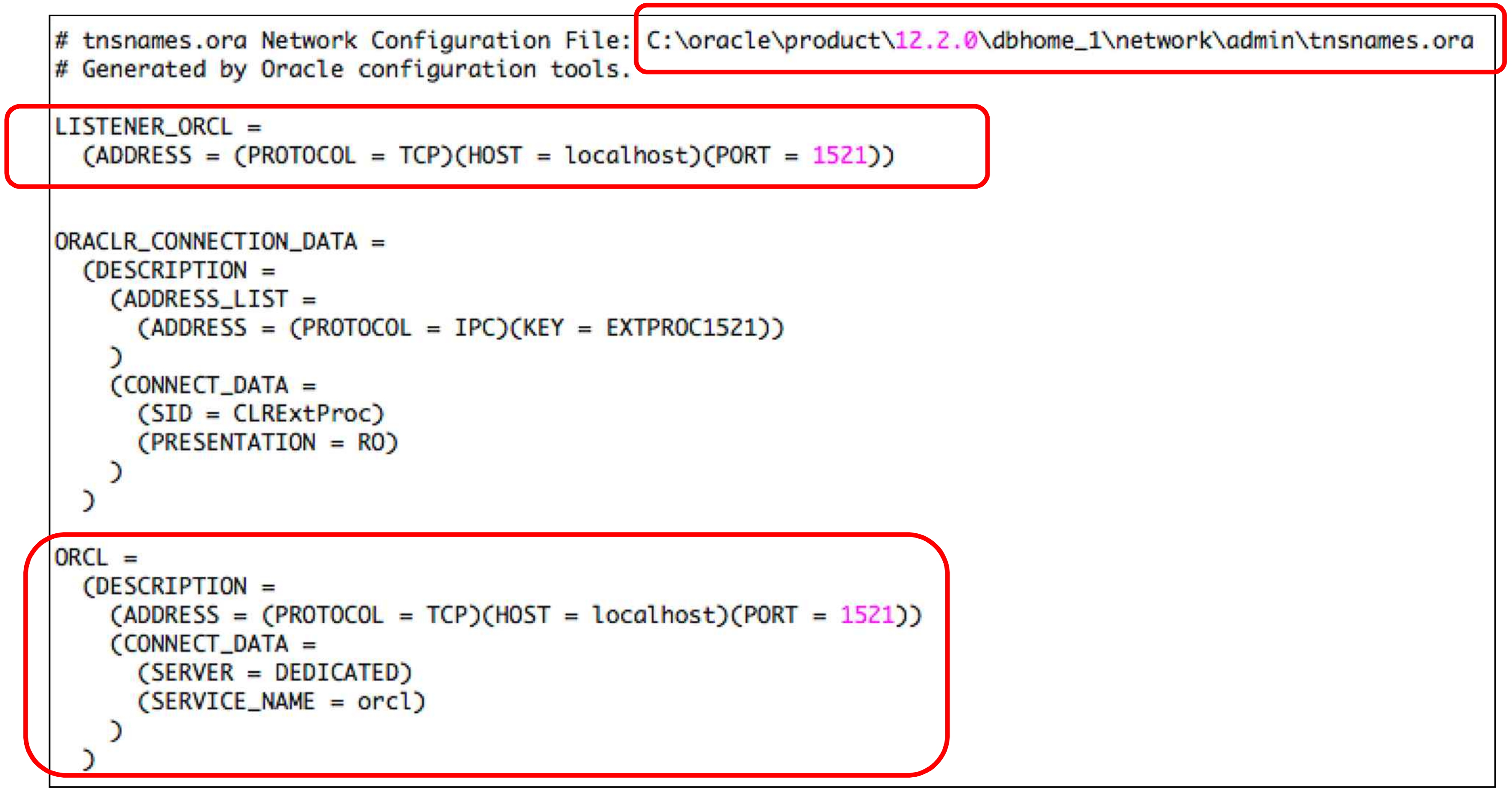
- Deadline: **Friday's midnight** (10/12/2018)
- Task: Write a JDBC program to repeat the entire Lab #5.
  - Utilize the explained code (`TestJDBC.java`) under copyright on LMS.
  - [Important] The DDL used in Lab #5 can be hard-wired into your program, but you are required to load into `COMPANY` the tuples that your program **MUST** read from `company.txt` provided on LMS.
    - Read each line, identify table name or a tuple, and then insert records into a corresponding table.
    - This indicates requiring you to **read** the file and populate the database via **Java file I/O**.
- Submission:
  - Name your file as '`lab6-`Your_Student_ID`.java`'.
  - Upload it into LMS.

APPENDIX

Troubleshooting

Oracle Connection Error

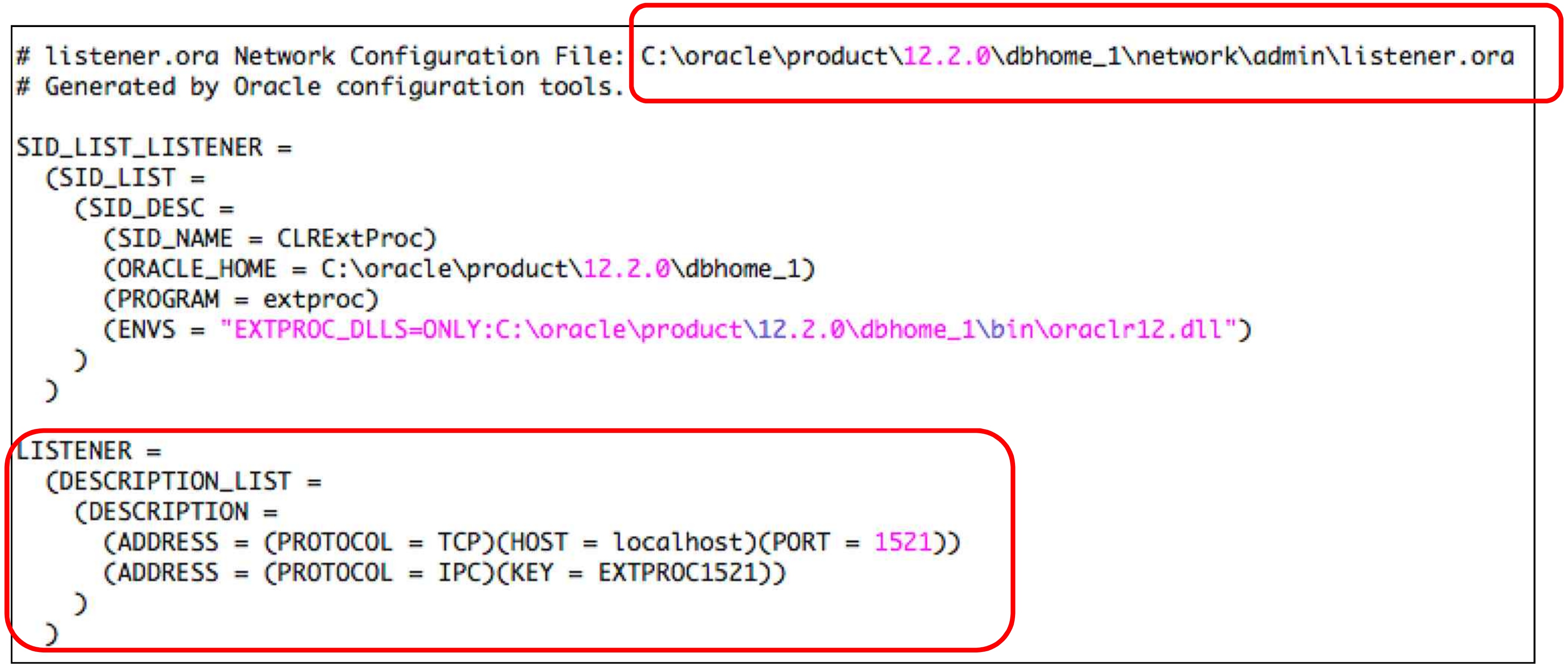
- Solution 1: Check your `tnsnames.ora` file



```
# tnsnames.ora Network Configuration File: C:\oracle\product\12.2.0\dbhome_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

LISTENER_ORCL =
  (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))


ORACLR_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
    (CONNECT_DATA =
      (SID = CLRExtProc)
      (PRESENTATION = RO)
    )
  )

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )
```

Oracle Connection Error (Cont'd)

- Solution 2: Check your `listener.ora` file



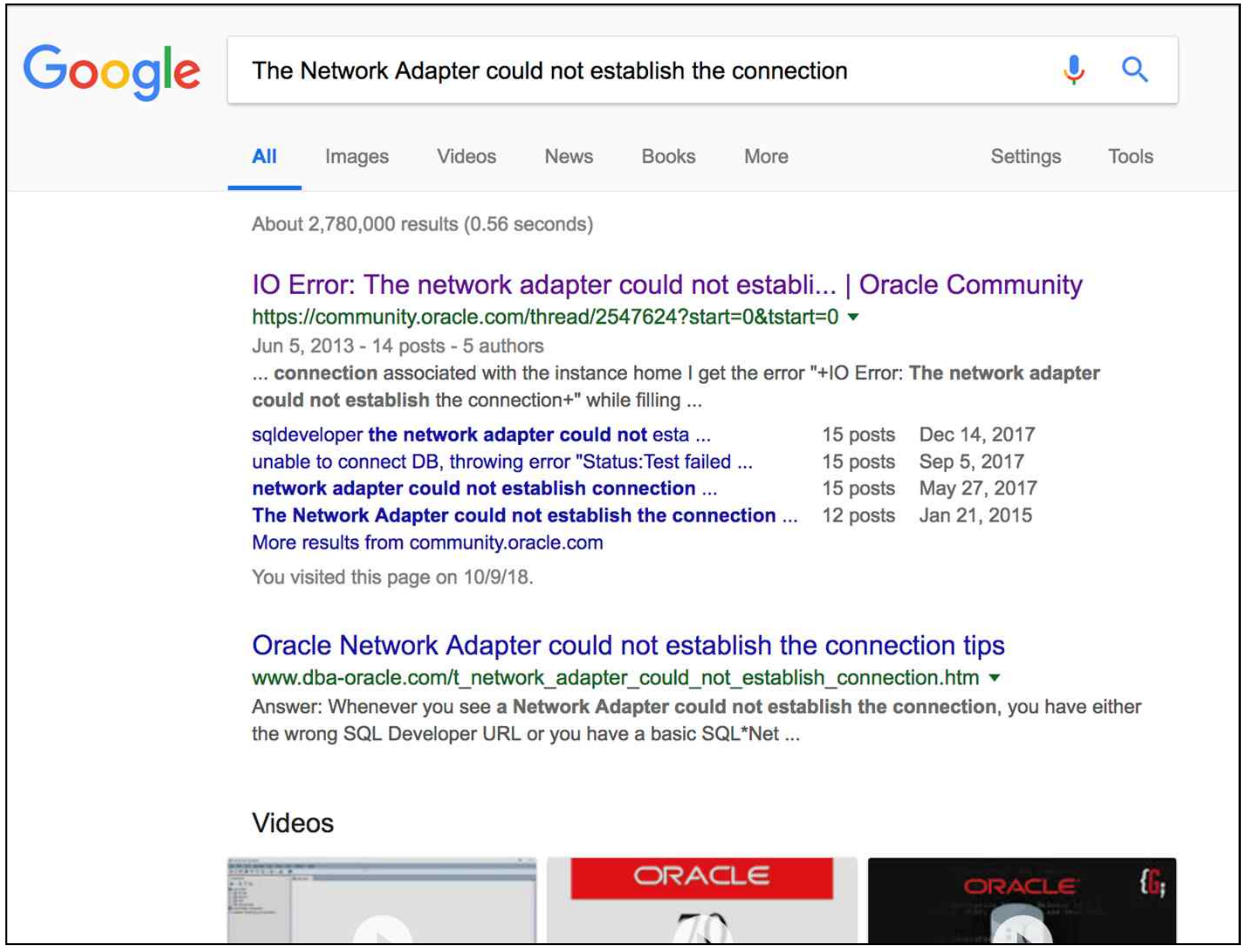
```
# listener.ora Network Configuration File: C:\oracle\product\12.2.0\dbhome_1\network\admin\listener.ora
# Generated by Oracle configuration tools.

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = CLRExtProc)
      (ORACLE_HOME = C:\oracle\product\12.2.0\dbhome_1)
      (PROGRAM = extproc)
      (ENVS = "EXTPROC_DLLS=ONLY:C:\oracle\product\12.2.0\dbhome_1\bin\oraclr12.dll")
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )
```

Oracle Connection Error (Cont'd)

- Solution 2: Open Windows Firewall
  - Check this website:
    - http://javadeveloper.tistory.com/m/7

Oracle Connection Error (Cont'd)

- Solution 3: Google is your friend.



About 2,780,000 results (0.56 seconds)

IO Error: The network adapter could not establi... | Oracle Community
https://community.oracle.com/thread/2547624?start=0&tstart=0
Jun 5, 2013 - 14 posts - 5 authors
... **connection** associated with the instance home I get the error "+IO Error: **The network adapter could not establish** the connection+" while filling ...

sqldeveloper **the network adapter could not** esta ...	15 posts	Dec 14, 2017
unable to connect DB, throwing error "Status:Test failed ...	15 posts	Sep 5, 2017
network adapter could not establish connection ...	15 posts	May 27, 2017
The Network Adapter could not establish the connection ...	12 posts	Jan 21, 2015

More results from community.oracle.com
You visited this page on 10/9/18.

Oracle Network Adapter could not establish the connection tips
www.dba-oracle.com/t_network_adapter_could_not_establish_connection.htm
Answer: Whenever you see a **Network Adapter could not establish the connection**, you have either the wrong SQL Developer URL or you have a basic SQL*Net ...

Videos