JDBC with Eclipse

LIAN LIU

DEPARTMENT OF COMPUTER SCIENCE

USC

Installation

Install JDK

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Install Oracle (optional)

http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html

Install Eclipse (Classic)

http://www.eclipse.org/downloads/

Download JDBC drivers

http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html

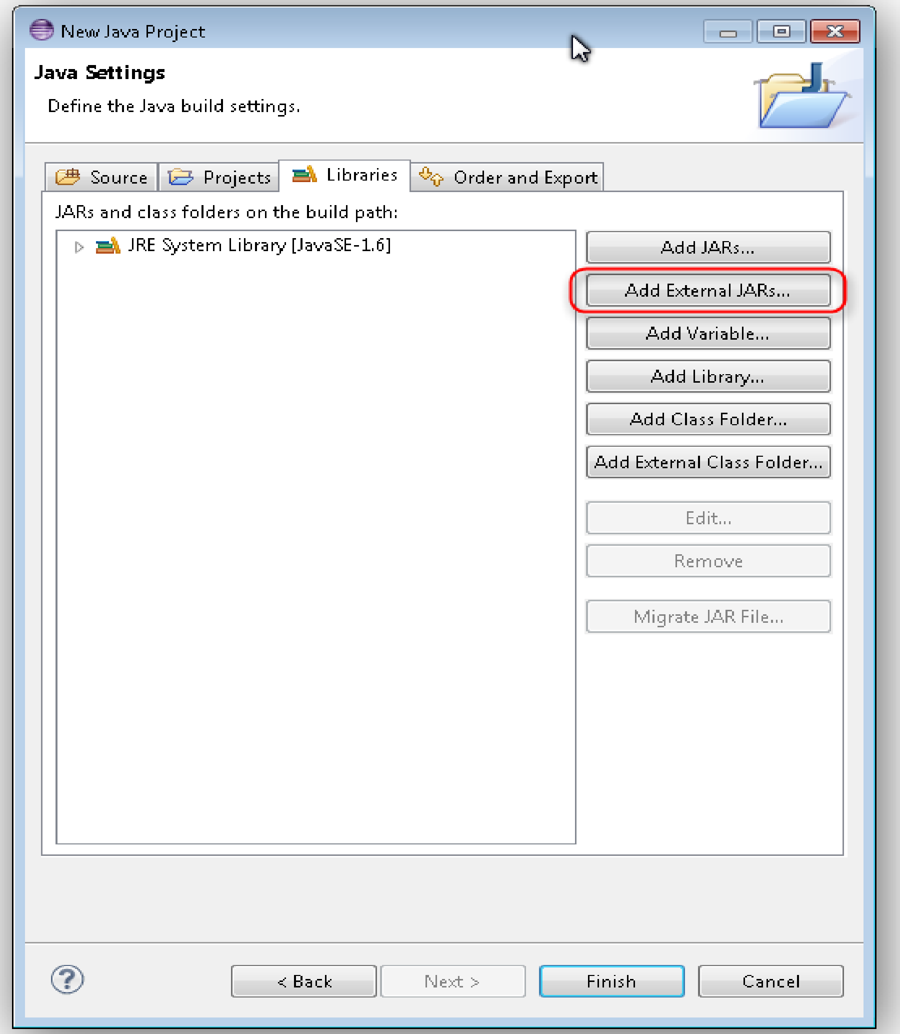
http://www.oracle.com/technetwork/database-options/spatialandgraph/downloads/software/sp-download-distlic-522138.html

Check your Oracle and Java version

- Check your Oracle version using the following command under sqlplus:
  - sqlplus
  - user: system
  - password: ******
  - select * from v$version
  - where banner like 'Oracle%';
- Check your Java version using the following command under Command Prompt:
  - java -version

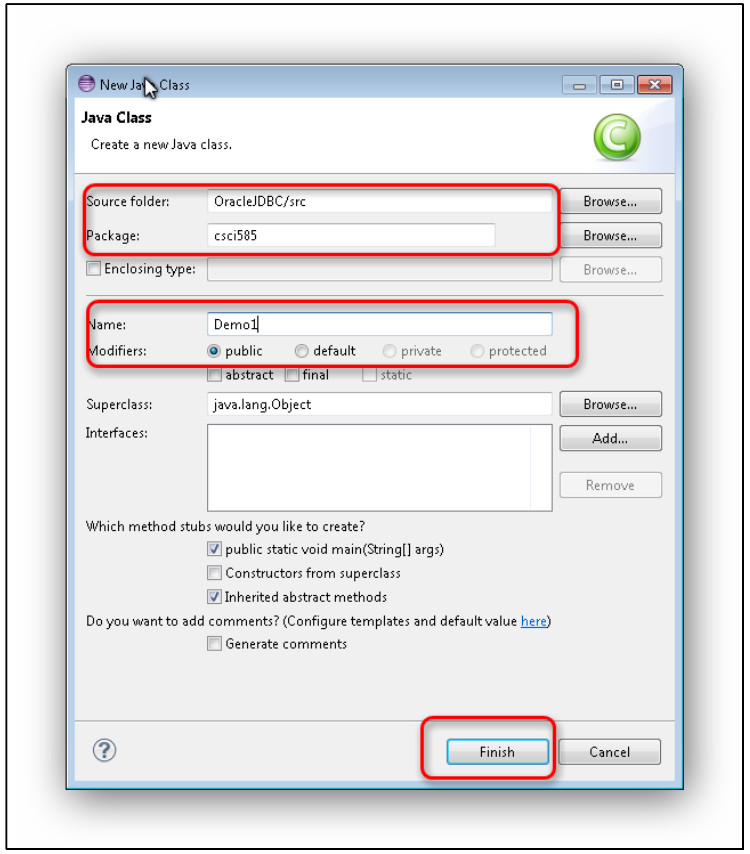
Creating a new Java project

- Create a new Java project by selecting "**File** → **New** → **Project**.. →**Java project**, (**Next**)"
- Specify a project name (e.g., JDBCTest), (**Next**)
- In Java settings dialog, select "**Libraries** → **Add external JARs**", and then select "**ojdbc6.jar**", "**sdoapi.jar**" and "**sdoutl.jar**" from the folder where you have downloaded them. (**Finish**)



Creating a new class

- Add a new class by right click “src” in Package Explorer on the left side, select “**new → class**”
- Specify the package name (e.g., csci585) and the class name (“Demo1”)



New Java Class

Java Class

Create a new Java class.

Source folder: OracleJDBC/src

Package: csci585

[ ] Enclosing type:

Name: Demo1

Modifiers: [✓ public] default private protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[✓] public static void main(String[] args)

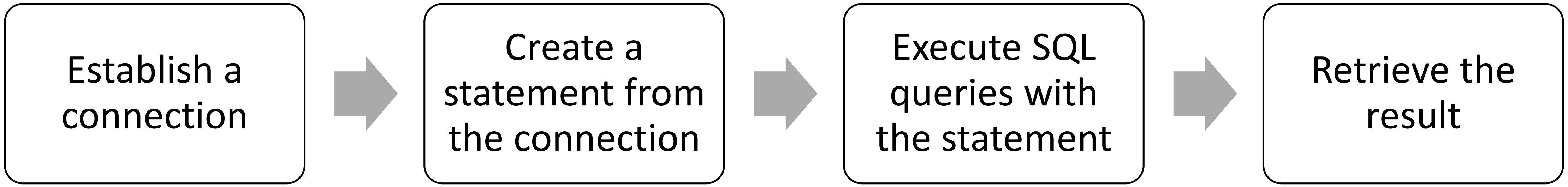
[ ] Constructors from superclass

[✓] Inherited abstract methods

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

[ ] Generate comments

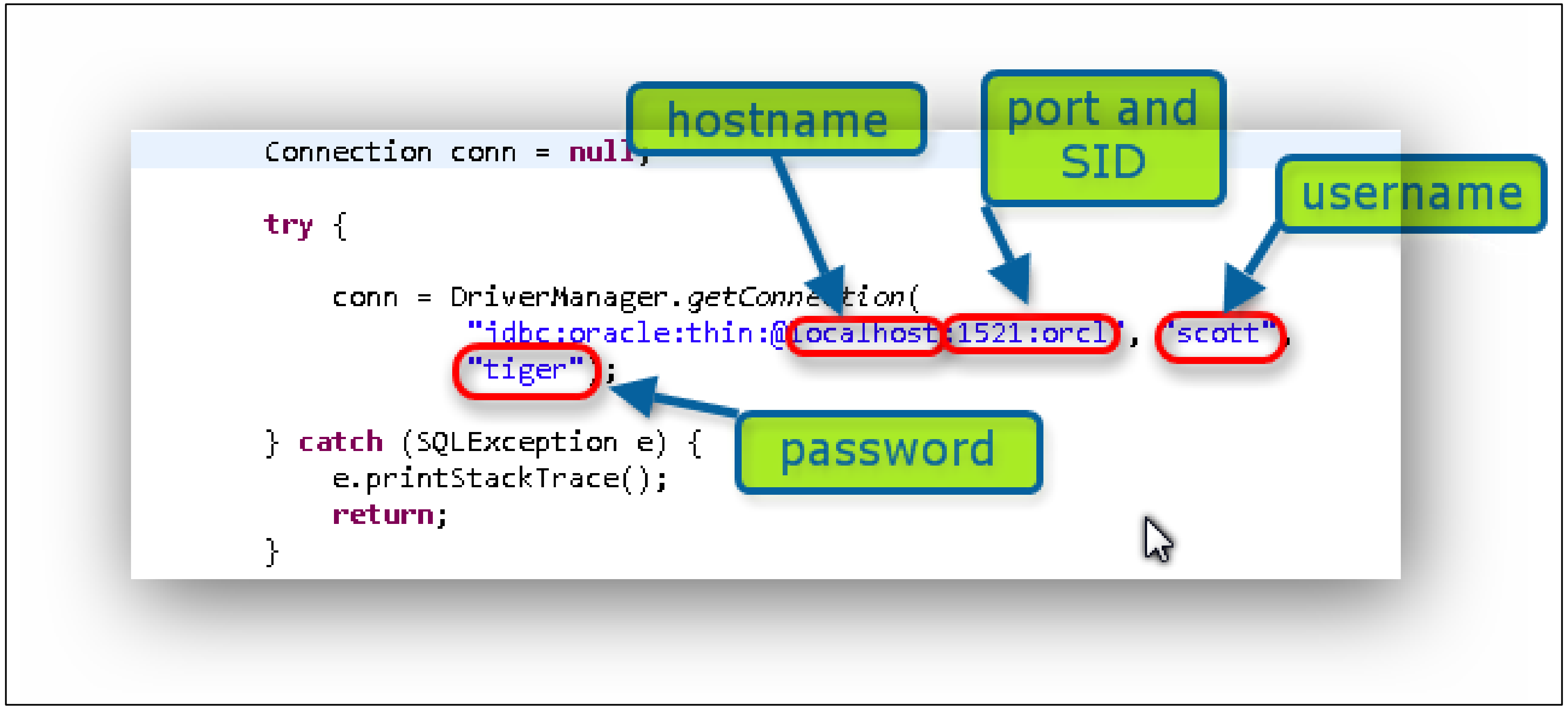
An overview of the process


Establish a connection
Create a statement from the connection
Execute SQL queries with the statement
Retrieve the result

Establishing a DB connection

- A connection is an object that serves as a communication bridge between Java programs and databases
- A database connection can be established using the following method:
- **Connection DriverManager.getConnection(String)**
  - **Input: Connection description string**
  - **Output: A connection object to the database (or null if connection fails)**
- Example:
- **Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "scott", "tiger");**
- A connection must be closed when it is no longer needed: **conn.close();**

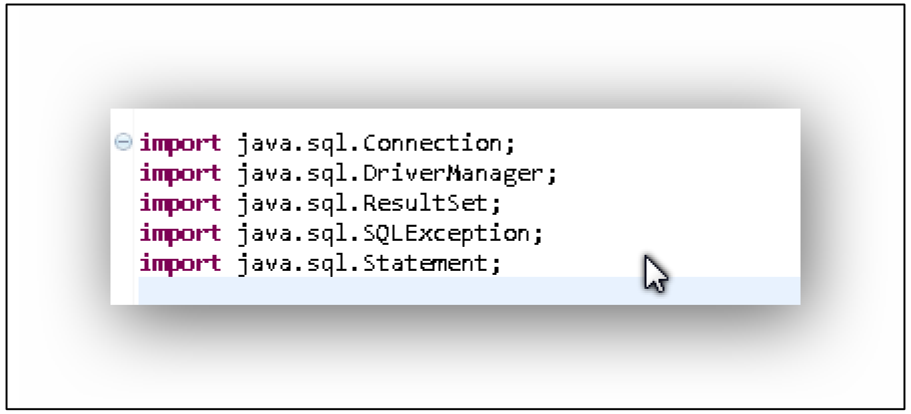
Establishing a DB connection



```
    Connection conn = null;

    try {

        conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:orcl", "scott",
                "tiger");

    } catch (SQLException e) {
        e.printStackTrace();
        return;
    }
```

Importing required packages

If the compiler cannot resolve the type name, maybe it is because some required packages are missing. In this case, you may use the Eclipse short-cut "Ctrl-Shift-O" to organize imports automatically.



```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

Creating a statement

- A statement is an object that carries SQL queries that can be executed on database
- A statement can be created using the createStatement method of an Connection object:
- **Statement conn.createStatement()**
  - **Input: void**
  - **Output: a statement object**
  - Here conn is a connection object
- Example:
- **Statement stmt = conn.createStatement();**
- A statement must be closed once it is no longer needed: **stmt.close();**

Executing queries

- An SQL query can be executed using the executeQuery method of a Statement object:
- **ResultSet stmt.executeQuery(String)**
  - **Input: a SQL query string**
  - **Output: a ResultSet object**
  - Here stmt is a statement object
- Example:
- **String query = "SELECT * FROM DEPT";**
- **ResultSet rs = stmt.executeQuery(query);**

Accessing the query result

- The query results can be returned as ResultSet objects
- We can access each row using a while loop as follows:
  - while (rs.next()) {
    - int deptno = rs.getInt("DEPTNO");
    - String dname = rs.getString("DNAME");
    - String loc = rs.getString("LOC");
    - System.out.println(deptno + " " + dname + " " + loc);
  - }

A whole JDBC example

- The following is a JDBC example on the web (tutorialspoint.com):
- http://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm

Spatial queries

- You can run spatial queries in exactly the same way as normal queries
- Example:
  - String query = “CREATE TABLE STUDENT
  - (ID NUMBER NOT NULL , NAME VARCHAR2(20) NOT NULL , GENDER VARCHAR2(20) NOT NULL
  - , HEIGHT NUMBER NOT NULL , DOB DATE NOT NULL , LOC MDSYS.SDO_GEOMETRY NOT NULL
  - , CONSTRAINT STUDENT_PK PRIMARY KEY   (ID   ) ENABLE);
  - ResultSet rs = stmt.executeQuery(query);

Reference

For query syntax & examples for spatial queries, you may refer to the Oracle official guide:

http://docs.oracle.com/cd/B28359_01/appdev.111/b28400.pdf