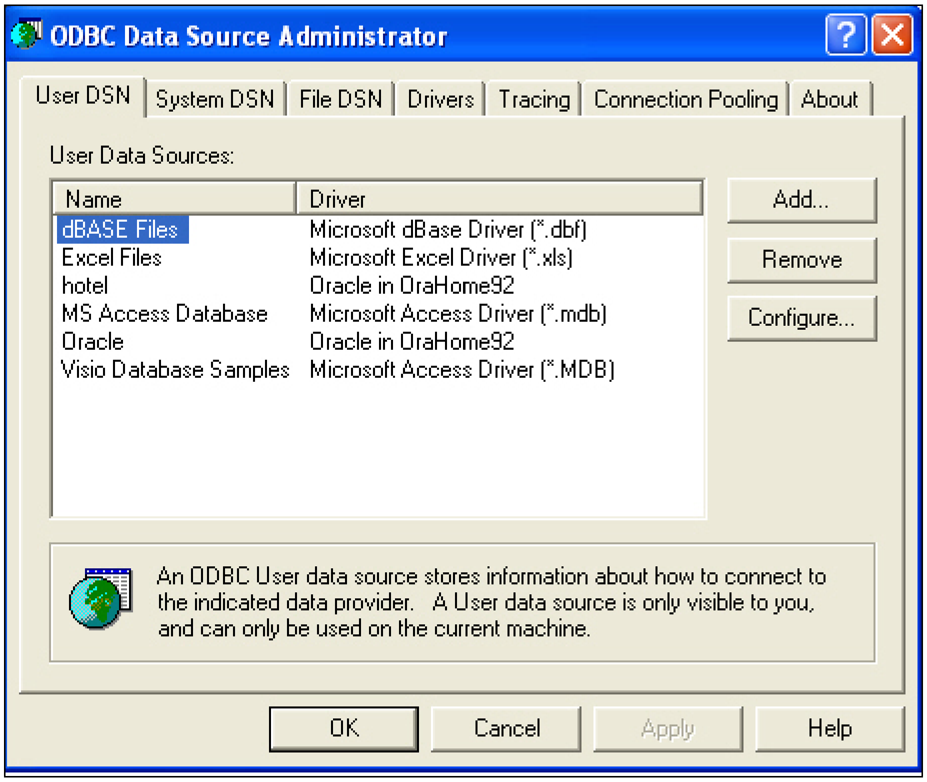
Lecture 6 Part 2 JDBC

Objectives

- Discuss setting up JDBC connectivity.
- Demonstrate a JDBC program
- Discuss and demonstrate methods associated with JDBC connectivity

Setting Up JDBC

Before you can begin to utilize JDBC, you must setup ODBC connectivity to the instance of Oracle that you are connecting to. This is done through the control panel. Depending upon the version of Windows that you are running, you may find the ODBC administrator under administrative tools or it may just be an option in the control panel. Figure 1 shows the main screen of the ODBC administrator.



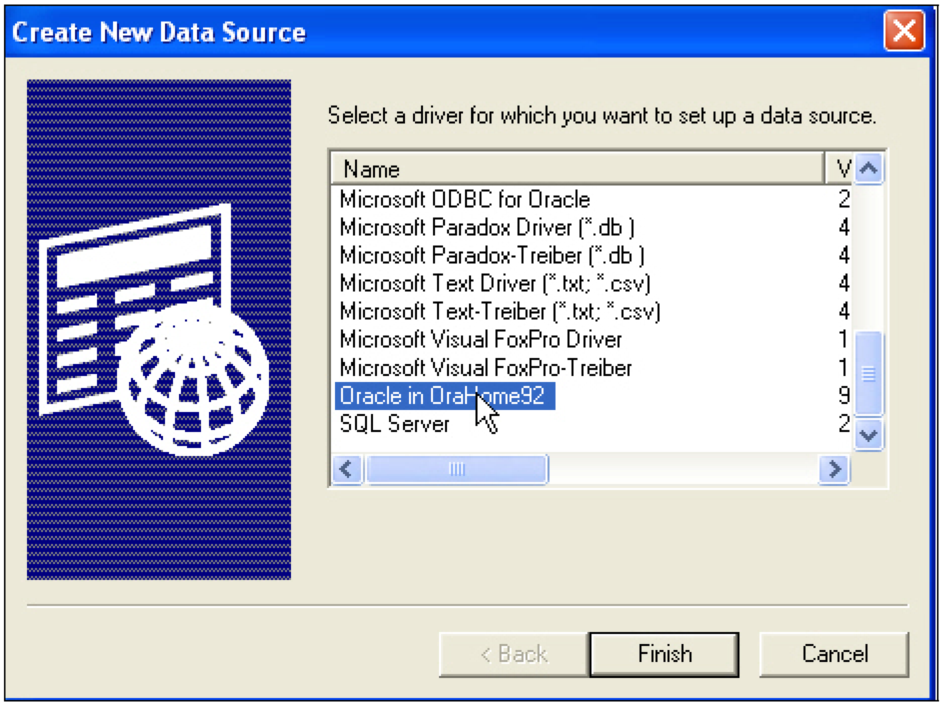
User Data Sources:

Name	Driver
dBASE Files	Microsoft dBase Driver (*.dbf)
Excel Files	Microsoft Excel Driver (*.xls)
hotel	Oracle in OraHome92
MS Access Database	Microsoft Access Driver (*.mdb)
Oracle	Oracle in OraHome92
Visio Database Samples	Microsoft Access Driver (*.MDB)

An ODBC User data source stores information about how to connect to the indicated data provider.   A User data source is only visible to you, and can only be used on the current machine.

Figure 1

I already have Oracle setup. If you do not, you will need to add a User DSN to connect to Oracle by clicking on Add. Once you do this, the screen shown in Figure 2 (or something similar) will appear:



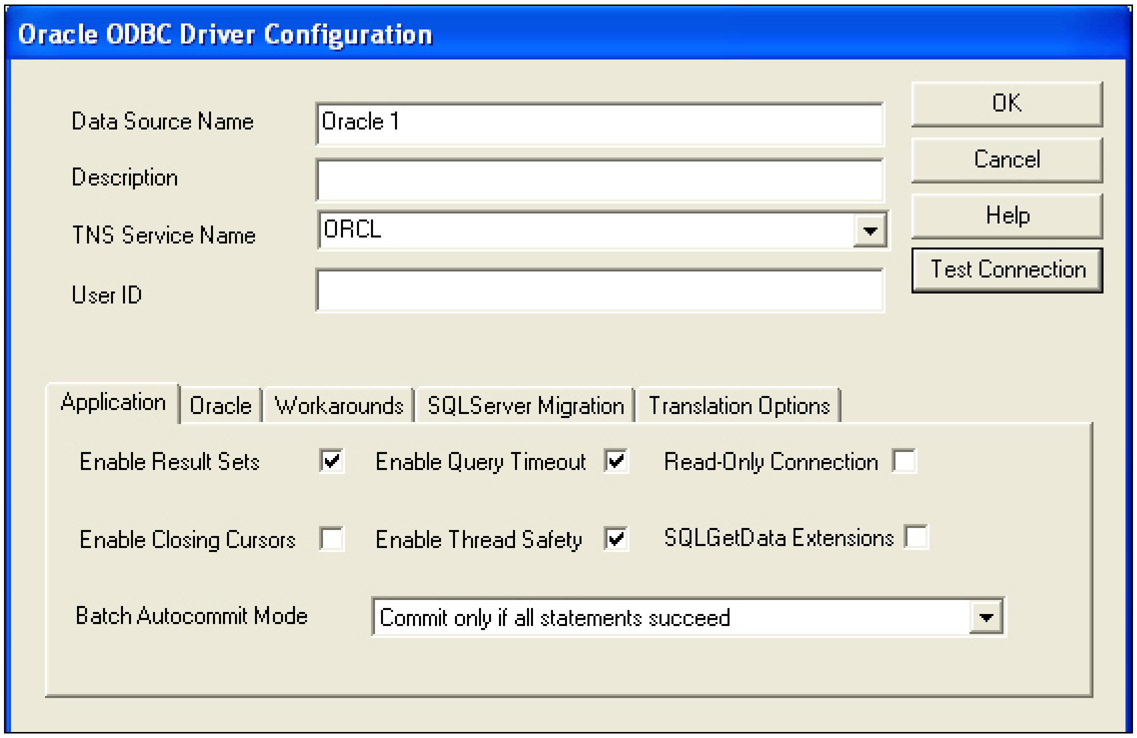
Create New Data Source

Select a driver for which you want to set up a data source.

Name	V
Microsoft ODBC for Oracle	2
Microsoft Paradox Driver (*.db)	4
Microsoft Paradox-Treiber (*.db)	4
Microsoft Text Driver (*.txt; *.csv)	4
Microsoft Text-Treiber (*.txt; *.csv)	4
Microsoft Visual FoxPro Driver	1
Microsoft Visual FoxPro-Treiber	1
Oracle in OraHome92	9
SQL Server	2

Figure 2

Please be sure that you select Oracle in OraHome92 and that the driver is for Oracle 9.2. For Oracle 10g, it will be Oracle in OraHome10g. Click on Finish. A screen similar to Figure 3 should now appear.



Oracle ODBC Driver Configuration

Data Source Name	Oracle 1
Description	
TNS Service Name	ORCL
User ID	

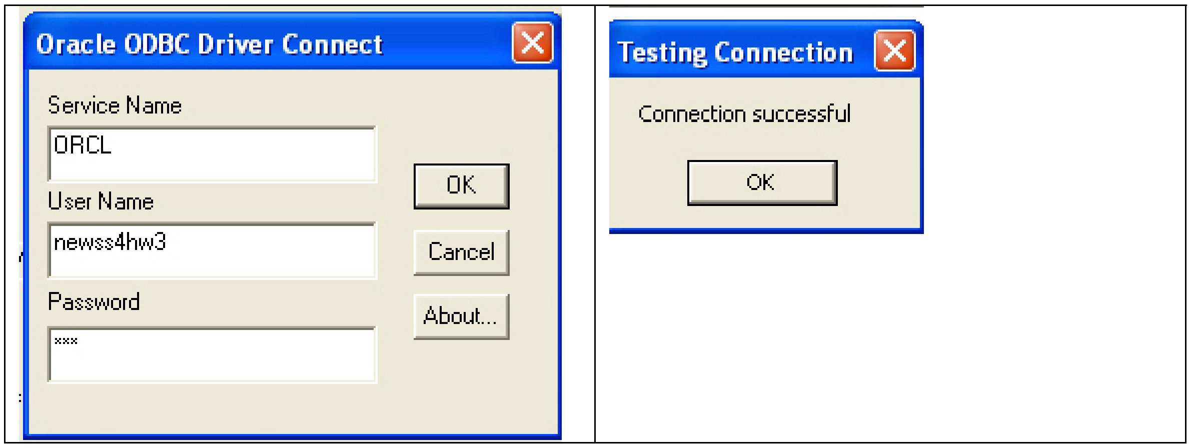
Enable Result Sets ✓ Enable Query Timeout ✓ Read-Only Connection ☐

Enable Closing Cursors ☐ Enable Thread Safety ✓ SQLGetData Extensions ☐

Batch Autocommit Mode: Commit only if all statements succeed

Figure 3

You should specify ORCL as the TNS Service name, unless you are connecting to another instance of Oracle through your employer (in that case, check with your DBA.) You should name the data source name Oracle or something similar. Once you have entered this information, click on Test Connection. A screen similar to Figure 4 will appear. Enter a valid username and password. If the connection is successful, you will get a message indicating this.



Service Name: ORCL
User Name: newss4hw3
Password: ***

Connection successful

Figure 4

Remember the name of the DSN, since this is what you will use to reference tables in your JDBC programs.

Install J2SE

You should install on your PC the J2SE 5.0 SDK. This can be downloaded from http://java.sun.com/j2se/1.5.0/download.jsp

Make sure that you select the JDK. If you already have an IDE for Java installed, you can use this.

Textpad – a handy tool for Java development

If you are not using an IDE for Java, a handy tool is Textpad, which is available for download at www.textpad.com You can compile and run your Java applications and applets directly from Textpad.

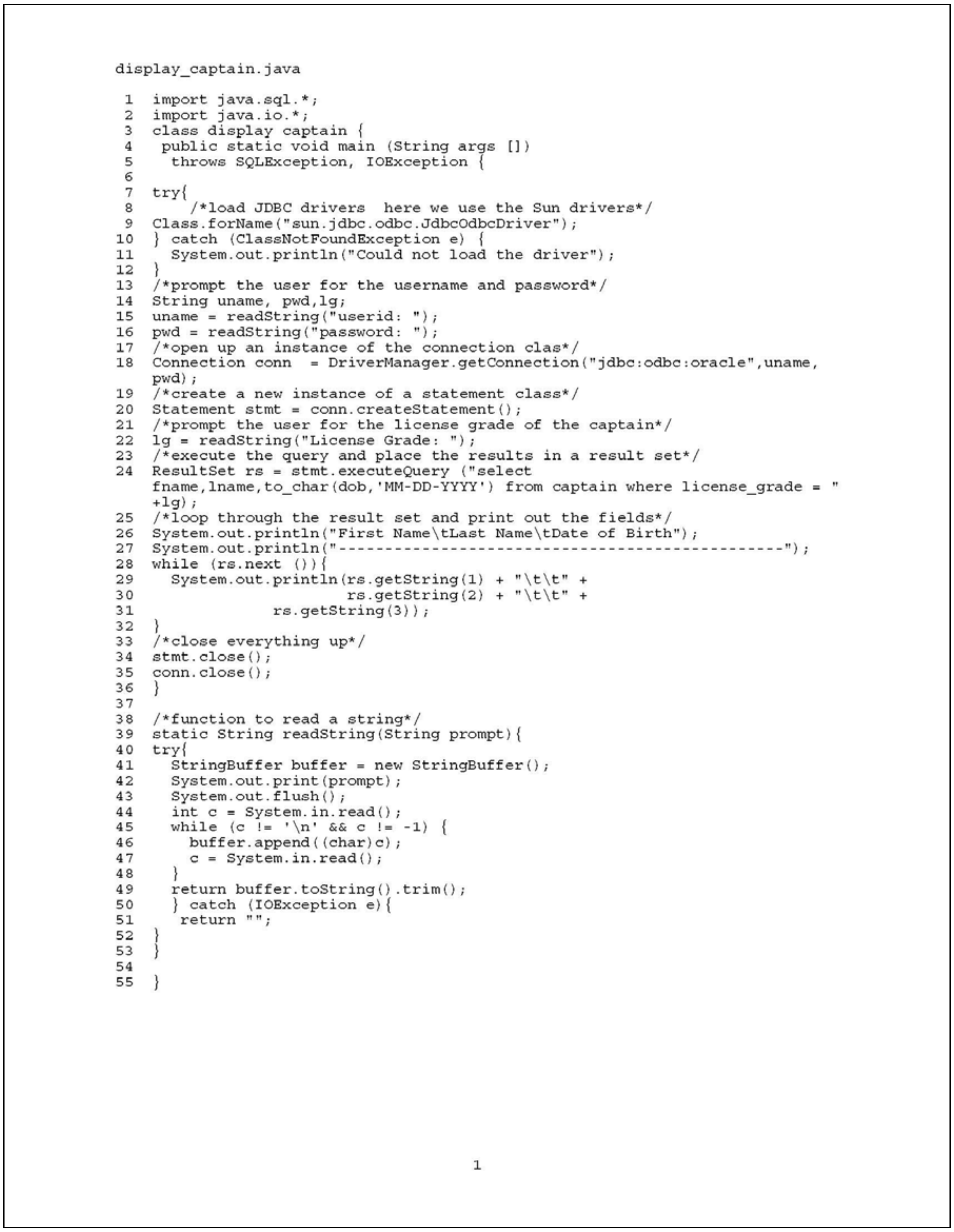
Java Tutorial

If you are new to Java, it would be worth your time to look at a tutorial on-line. One such tutorial is available at http://www.ibiblio.org/javafaq/javatutorial.html#xtocid90007. Ignore the instructions about installing Java.

JDBC – an Introduction

Java Database Connectivity or JDBC, is an API which consists of classes written in Java for connecting to a database. While similar to ODBC, it was developed especially for Java and provides a means to access databases independent of the database vendor and platform.

Figure 6 contains a simple JDBC application using our Shore to Shore shipping case study.



display_captain.java

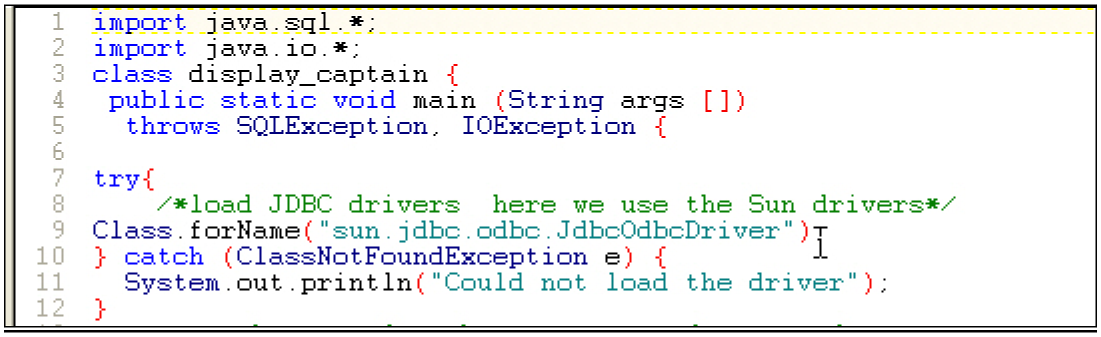
```
import java.sql.*;
import java.io.*;
class display captain {
 public static void main (String args [])
  throws SQLException, IOException {

try{
    /*load JDBC drivers  here we use the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
  System.out.println("Could not load the driver");
}
/*prompt the user for the username and password*/
String uname, pwd,lg;
uname = readString("userid: ");
pwd = readString("password: ");
/*open up an instance of the connection clas*/
Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
pwd);
/*create a new instance of a statement class*/
Statement stmt = conn.createStatement();
/*prompt the user for the license grade of the captain*/
lg = readString("License Grade: ");
/*execute the query and place the results in a result set*/
ResultSet rs = stmt.executeQuery ("select
fname,lname,to_char(dob,'MM-DD-YYYY') from captain where license_grade = "
+lg);
/*loop through the result set and print out the fields*/
System.out.println("First Name\tLast Name\tDate of Birth");
System.out.println("---------------------------------------------");
while (rs.next ()){
  System.out.println(rs.getString(1) + "\t\t" +
                     rs.getString(2) + "\t\t" +
             rs.getString(3));
}
/*close everything up*/
stmt.close();
conn.close();
}

/*function to read a string*/
static String readString(String prompt){
try{
  StringBuffer buffer = new StringBuffer();
  System.out.print(prompt);
  System.out.flush();
  int c = System.in.read();
  while (c != '\n' && c != -1) {
    buffer.append((char)c);
    c = System.in.read();
  }
  return buffer.toString().trim();
  } catch (IOException e){
   return "";
}
}

}
```

1

Figure 6 (display_captain.java)

If you use Textpad, you can compile this directly by pressing the ctrl-1 key combination. The ctrl-2 key runs the application in an MS-DOS window. Let's look at each portion of the code.

Setting up the Driver

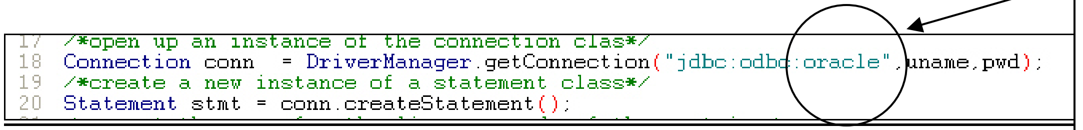


```
1  import java.sql.*;
2  import java.io.*;
3  class display_captain {
4   public static void main (String args [])
5    throws SQLException, IOException {
6
7  try{
8      /*load JDBC drivers  here we use the Sun drivers*/
9  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
10 } catch (ClassNotFoundException e) {
11   System.out.println("Could not load the driver");
12 }
```

Lines 1 and 2 are simply importing libraries into our Java program. We must have java.sql.* in order to use JDBC. For those of you who are not familiar with Java, line 3 defines our class and line 4 the main function on the class. We then call the jdbc drivers in line 9 using the Sun drivers (you can also use Microsoft drivers.) Notice that we "wrap" line 9 in a try/catch. This is Java's version of exception handling.

Lines 14-16 prompt the user for the Oracle username and password and store these to string variables. You could hardcode these values into your program or also pass them as arguments when you call the Java application.

Open the Connection Class

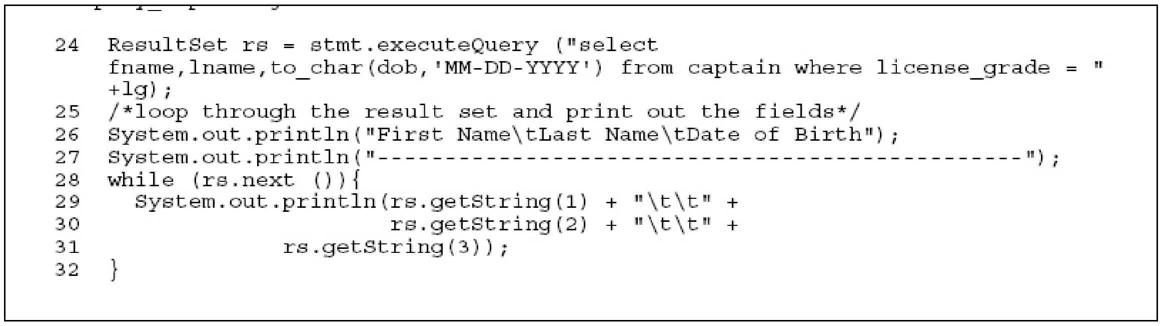


```
17 /*open up an instance of the connection clas*/
18 Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,pwd);
19 /*create a new instance of a statement class*/
20 Statement stmt = conn.createStatement();
```

The name of the DSN defined in ODBC configuration .

Line 18 opens up an instance of the connection class, using the driver specified in line 9. Notice that we specify the name of the DSN as well as the username and password. The DSN could be setup for any RDBMS, such as Oracle, SQL Server, MySQL, MS Access, etc. Line 20 creates a new instance of Statement using the createStatement method. There are three different types of statements, which we will discuss later.

Create a Resultset



```
24  ResultSet rs = stmt.executeQuery ("select
    fname,lname,to_char(dob,'MM-DD-YYYY') from captain where license_grade = "
    +lg);
25  /*loop through the result set and print out the fields*/
26  System.out.println("First Name\tLast Name\tDate of Birth");
27  System.out.println("---------------------------------------------");
28  while (rs.next ()){
29    System.out.println(rs.getString(1) + "\t\t" +
30                       rs.getString(2) + "\t\t" +
31                 rs.getString(3));
32  }
```

Line 24 creates an instance of a ResultSet called RS. We use the method executeQuery here to return some information from the captain table. This is similar to a cursor in PL/SQL. Notice in this line that we convert the date to a character string. Line 28 loops through the resultset using the next() method. rs.next() will return false when we have reached the end of the result set. The getString method returns columns in the resultset. In our example, we have three columns, so we use getString(1) for the firstname, getString(2) for the lastname and getString(3) for the date of birth. There are other methods for other types of data, which we will see in our next example.

readString Function

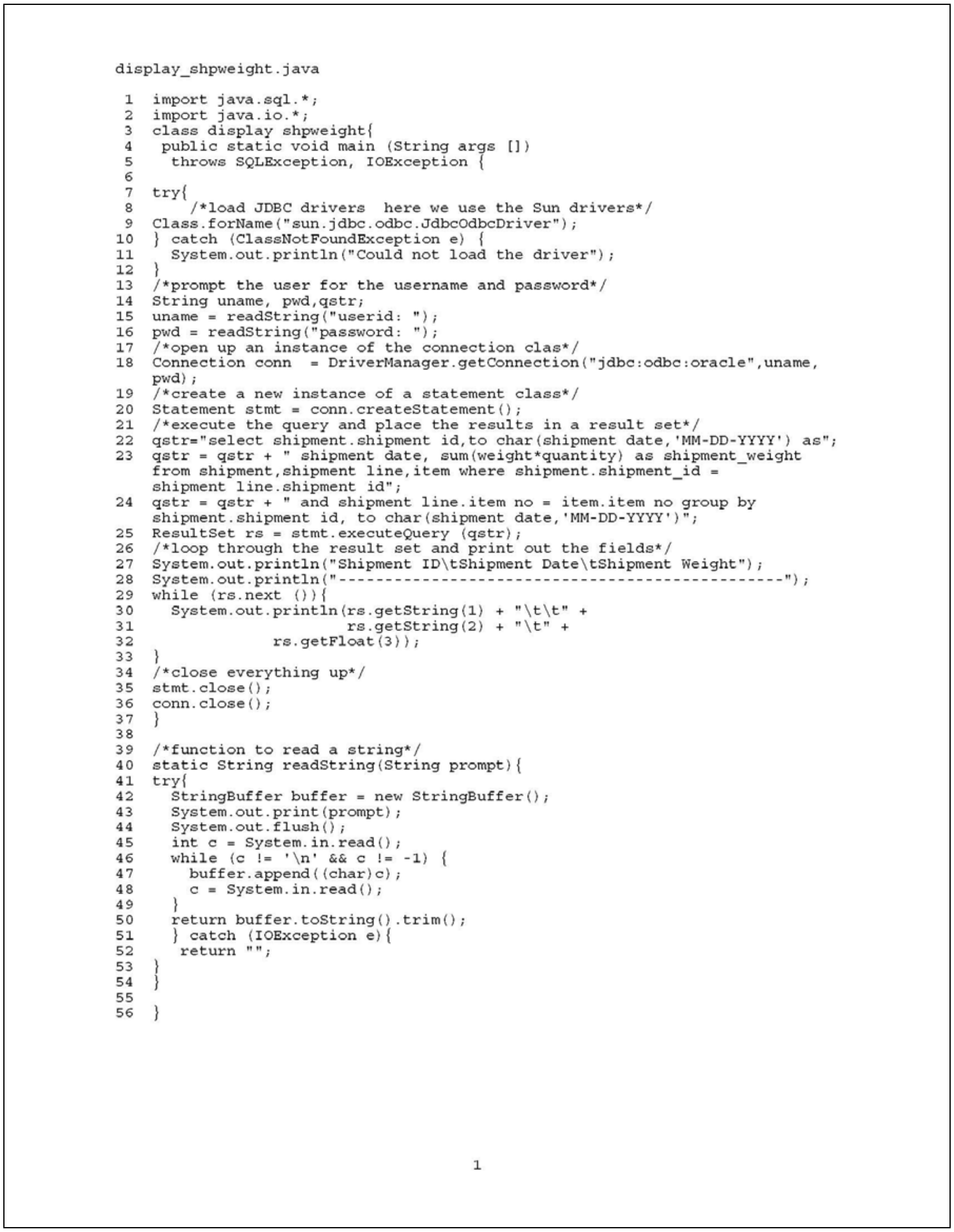
```
38  /*function to read a string*/
39  static String readString(String prompt){
40  try{
41    StringBuffer buffer = new StringBuffer();
42    System.out.print(prompt);
43    System.out.flush();
44    int c = System.in.read();
45    while (c != '\n' && c != -1) {
46      buffer.append((char)c);
47      c = System.in.read();
48    }
49    return buffer.toString().trim();
50    } catch (IOException e){
51     return "";
52  }
53  }
54
55  }
```

This function, which we will use in many of our examples, reads user input from the keyboard into a string using the StringBuffer class.

Example 1.1

Let's look at another example where we return information of different types. Let's use the following query, which returns the shipment_id, shipment_date and shipment_weight, for our ResultSet:

```
select shipment.shipment_id,to_char(shipment_date,'MM-DD-YYYY') as
        shipment_date, sum(weight*quantity) as shipment_weight
from shipment,shipment_line,item
where shipment.shipment_id = shipment_line.shipment_id
and shipment_line.item_no = item.item_no
group by shipment.shipment_id, to_char(shipment_date,'MM-DD-YYYY');
```



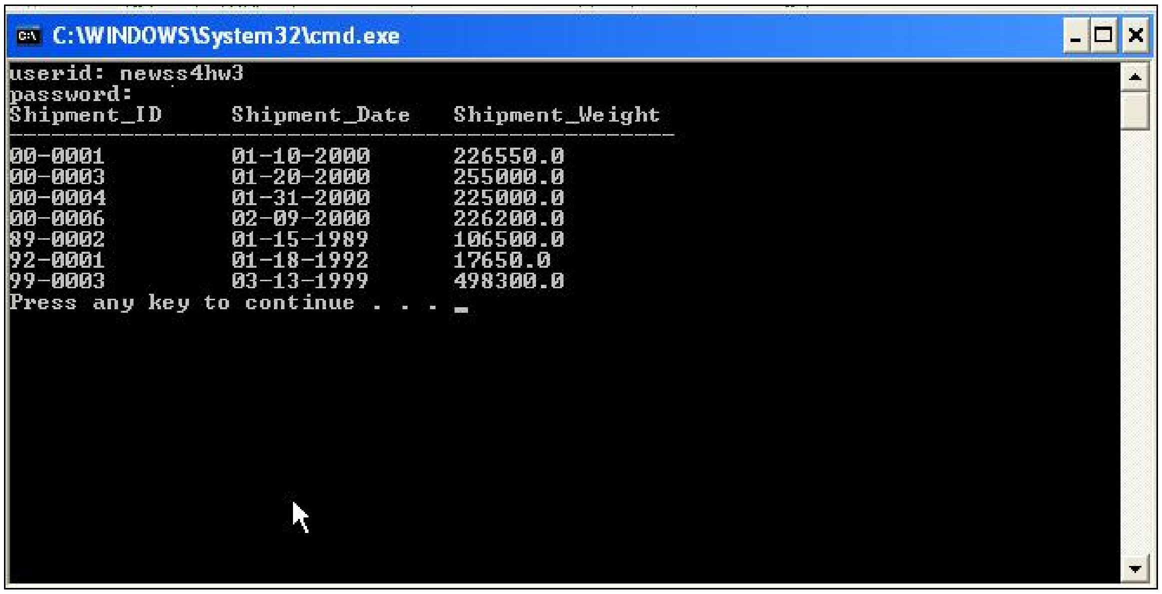
display_shpweight.java

```
 1  import java.sql.*;
 2  import java.io.*;
 3  class display shpweight{
 4   public static void main (String args [])
 5    throws SQLException, IOException {
 6
 7  try{
 8      /*load JDBC drivers  here we use the Sun drivers*/
 9  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
10  } catch (ClassNotFoundException e) {
11    System.out.println("Could not load the driver");
12  }
13  /*prompt the user for the username and password*/
14  String uname, pwd,qstr;
15  uname = readString("userid: ");
16  pwd = readString("password: ");
17  /*open up an instance of the connection clas*/
18  Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
    pwd);
19  /*create a new instance of a statement class*/
20  Statement stmt = conn.createStatement();
21  /*execute the query and place the results in a result set*/
22  qstr="select shipment.shipment id,to char(shipment date,'MM-DD-YYYY') as";
23  qstr = qstr + " shipment date, sum(weight*quantity) as shipment_weight
    from shipment,shipment line,item where shipment.shipment_id =
    shipment line.shipment id";
24  qstr = qstr + " and shipment line.item no = item.item no group by
    shipment.shipment id, to char(shipment date,'MM-DD-YYYY')";
25  ResultSet rs = stmt.executeQuery (qstr);
26  /*loop through the result set and print out the fields*/
27  System.out.println("Shipment ID\tShipment Date\tShipment Weight");
28  System.out.println("-----------------------------------------------");
29  while (rs.next ()){
30    System.out.println(rs.getString(1) + "\t\t" +
31                       rs.getString(2) + "\t" +
32              rs.getFloat(3));
33  }
34  /*close everything up*/
35  stmt.close();
36  conn.close();
37  }
38
39  /*function to read a string*/
40  static String readString(String prompt){
41  try{
42    StringBuffer buffer = new StringBuffer();
43    System.out.print(prompt);
44    System.out.flush();
45    int c = System.in.read();
46    while (c != '\n' && c != -1) {
47      buffer.append((char)c);
48      c = System.in.read();
49    }
50    return buffer.toString().trim();
51    } catch (IOException e){
52     return "";
53  }
54  }
55
56  }
```

1

Figure 7 display_shpweight.java

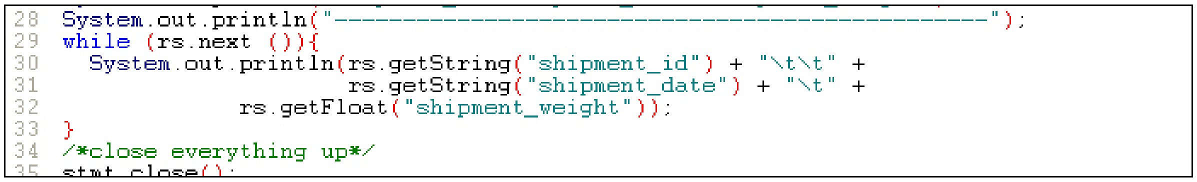
Notice that line 32 uses the getFloat method instead of getString since our third column returned is a number, The results returned from this program are shown in Figure 8.



```
userid: newss4hw3
password:
Shipment_ID      Shipment_Date    Shipment_Weight
------------------------------------------------
00-0001          01-10-2000       226550.0
00-0003          01-20-2000       255000.0
00-0004          01-31-2000       225000.0
00-0006          02-09-2000       226200.0
89-0002          01-15-1989       106500.0
92-0001          01-18-1992       17650.0
99-0003          03-13-1999       498300.0
Press any key to continue . . . _
```

Figure 8

We could have written this a bit differently, if we did not know the exact numbers of the columns that we wanted from the resultSet. In Figure 9, we use the name of the column instead of the number.



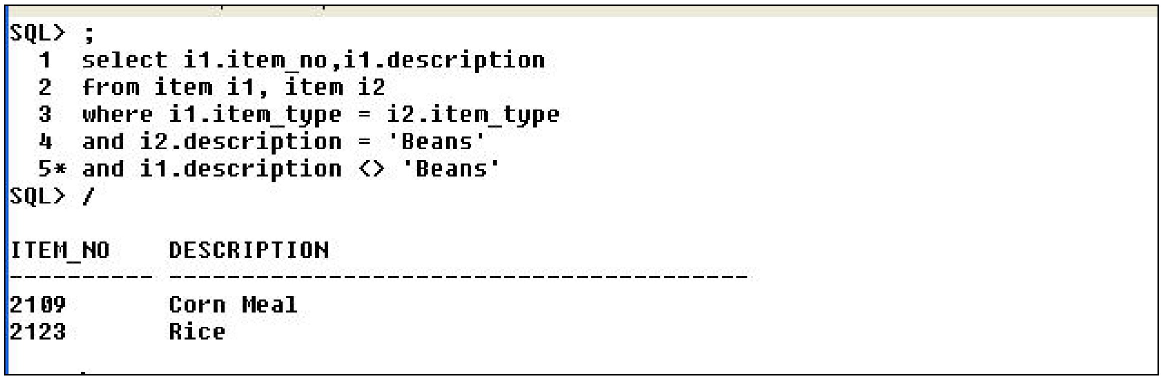
```
System.out.println("------------------------------------------------");
while (rs.next ()){
  System.out.println(rs.getString("shipment_id") + "\t\t" +
                     rs.getString("shipment_date") + "\t" +
              rs.getFloat("shipment_weight"));
}
/*close everything up*/
stmt.close();
```

Figure 9

On Your Own Exercises

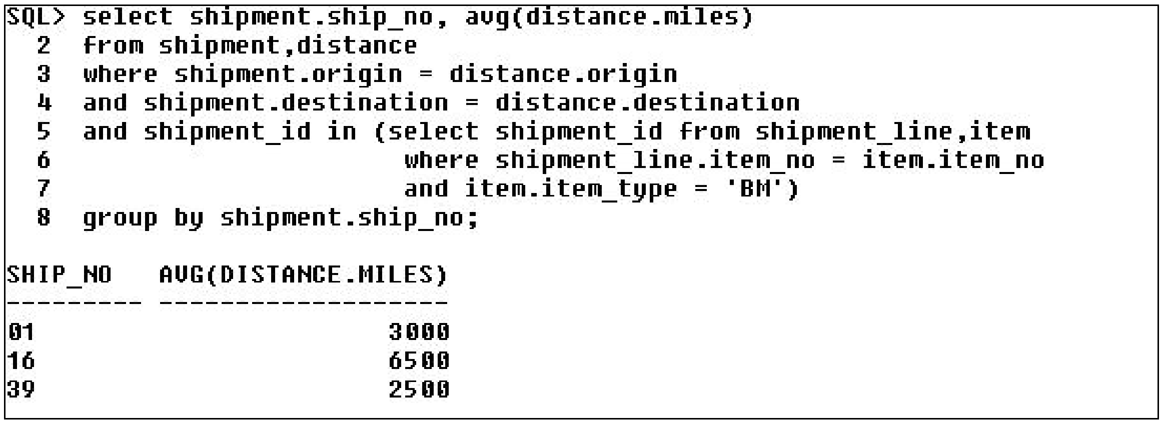
Write JDBC code to complete the following queries:

1.1 List item numbers and descriptions for all items that are of the same type as beans. (On Your Own Exercise 1.1 from Lecture 6 Part 1) The query is shown below.



```
SQL> ;
  1  select i1.item_no,i1.description
  2  from item i1, item i2
  3  where i1.item_type = i2.item_type
  4  and i2.description = 'Beans'
  5* and i1.description <> 'Beans'
SQL> /

ITEM_NO    DESCRIPTION
---------- ----------------------------------------
2109       Corn Meal
2123       Rice
```

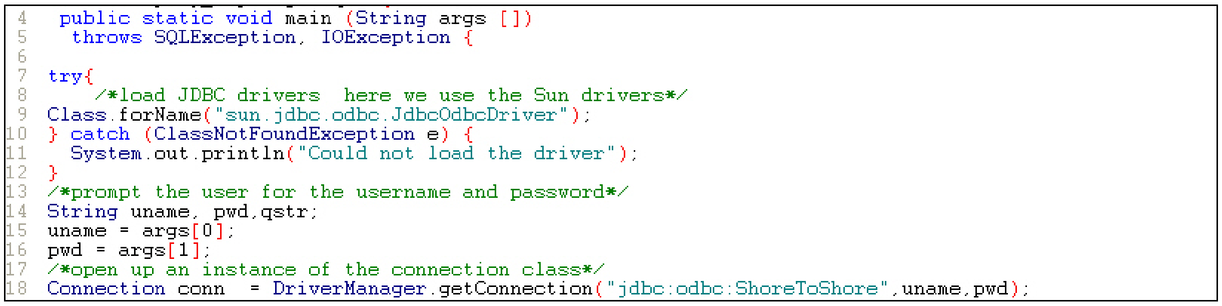
1.2 For each ship, list the average distance traveled for all shipments of building materials shipped on the ship.



```
SQL> select shipment.ship_no, avg(distance.miles)
  2  from shipment,distance
  3  where shipment.origin = distance.origin
  4  and shipment.destination = distance.destination
  5  and shipment_id in (select shipment_id from shipment_line,item
  6                      where shipment_line.item_no = item.item_no
  7                      and item.item_type = 'BM')
  8  group by shipment.ship_no;

SHIP_NO   AVG(DISTANCE.MILES)
--------- -------------------
01                       3000
16                       6500
39                       2500
```

Passing Login Information as Command Line Arguments

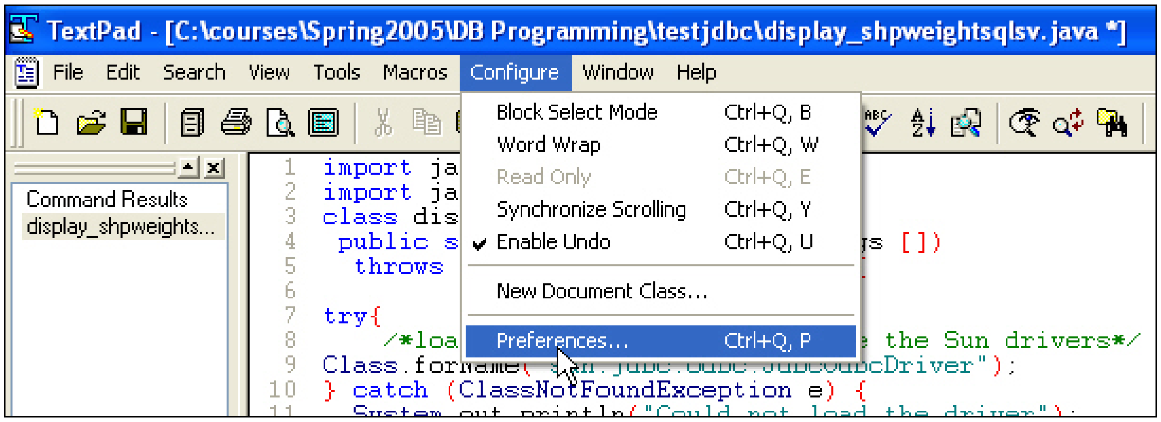
You might not always want to prompt the user for the login name and password. You can add the username to the DSN, however, you cannot add the password. Figure 10 shows lines 15 and 16 of example 1.1 changed to use parameters.



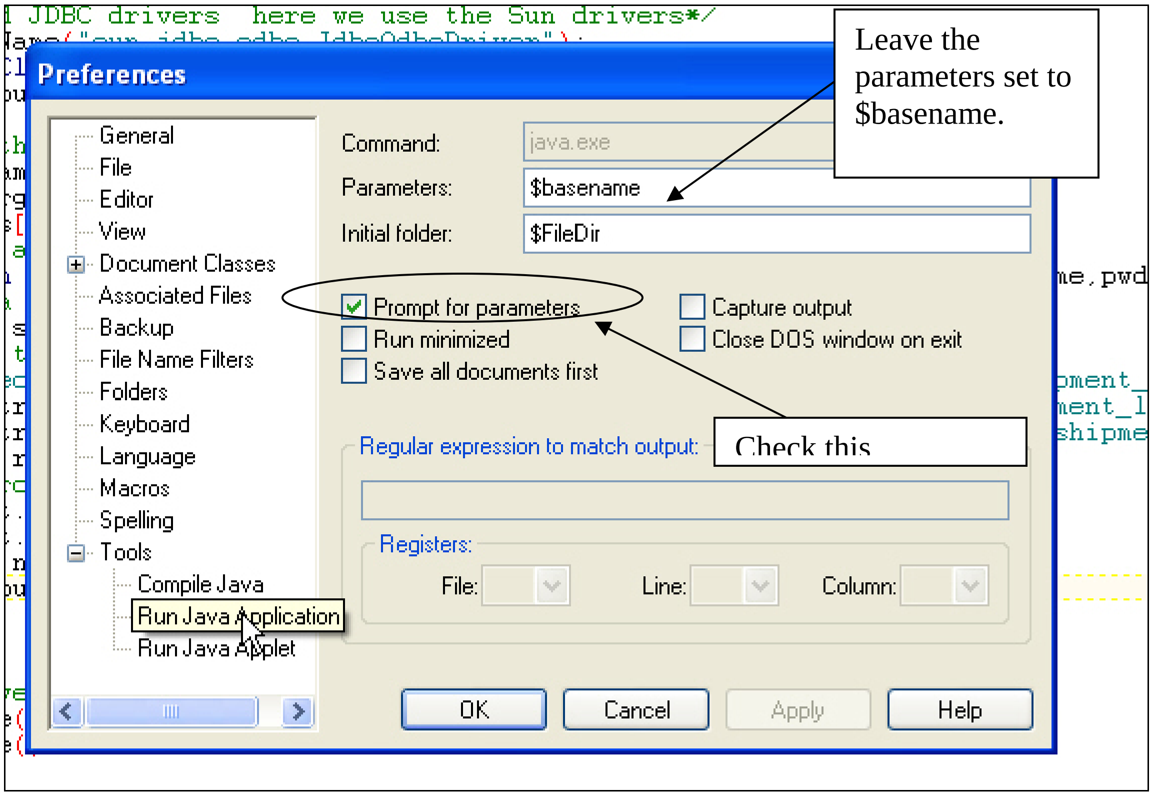
```
4   public static void main (String args [])
5    throws SQLException, IOException {
6
7  try{
8      /*load JDBC drivers  here we use the Sun drivers*/
9  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
10 } catch (ClassNotFoundException e) {
11   System.out.println("Could not load the driver");
12 }
13 /*prompt the user for the username and password*/
14 String uname, pwd,qstr;
15 uname = args[0];
16 pwd = args[1];
17 /*open up an instance of the connection class*/
18 Connection conn  = DriverManager.getConnection("jdbc:odbc:ShoreToShore",uname,pwd);
```

Figure 10

When you run this from the command prompt, you can simply enter the username and password as arguments. From within Textpad, you need to make sure that you set your preferences to prompt for parameters. The next few figures show the steps involved to do this.


TextPad - [C:\courses\Spring2005\DB Programming\testjdbc\display_shpweightsqlsv.java *]
File Edit Search View Tools Macros Configure Window Help
Block Select Mode Ctrl+Q, B
Word Wrap Ctrl+Q, W
Read Only Ctrl+Q, E
Synchronize Scrolling Ctrl+Q, Y
✓ Enable Undo Ctrl+Q, U
New Document Class...
Preferences... Ctrl+Q, P
Command Results
display_shpweights...

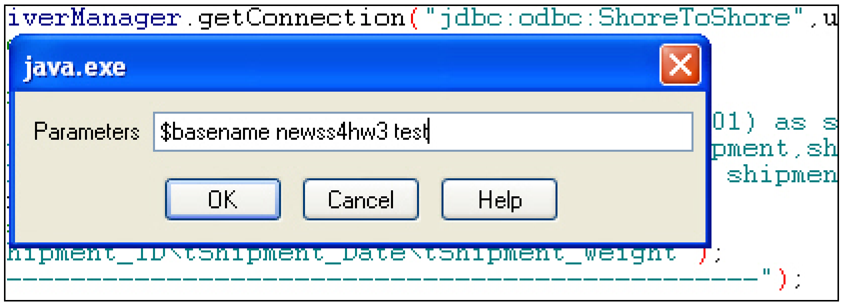
```
import ja
import ja
class dis
  public s                                   gs [])
    throws

try{
     /*loa                                 e the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
```



Command: java.exe
Parameters: $basename
Initial folder: $FileDir

[✓] Prompt for parameters
[ ] Run minimized
[ ] Save all documents first
[ ] Capture output
[ ] Close DOS window on exit

Once you have configured your preferences, you will then be prompted for the parameters. Leave $basename in the parameters box (this is the name of your class) and add your parameters after it (make sure that you have a space after $basename.)



```
iverManager.getConnection("jdbc:odbc:ShoreToShore",u
                                                   01) as s
                                                   pment,sh
                                                    shipmen
hipment_ID\tShipment_Date\tShipment_Weight");
----------------------------------------------------");
```

Parameters: $basename newss4hw3 test

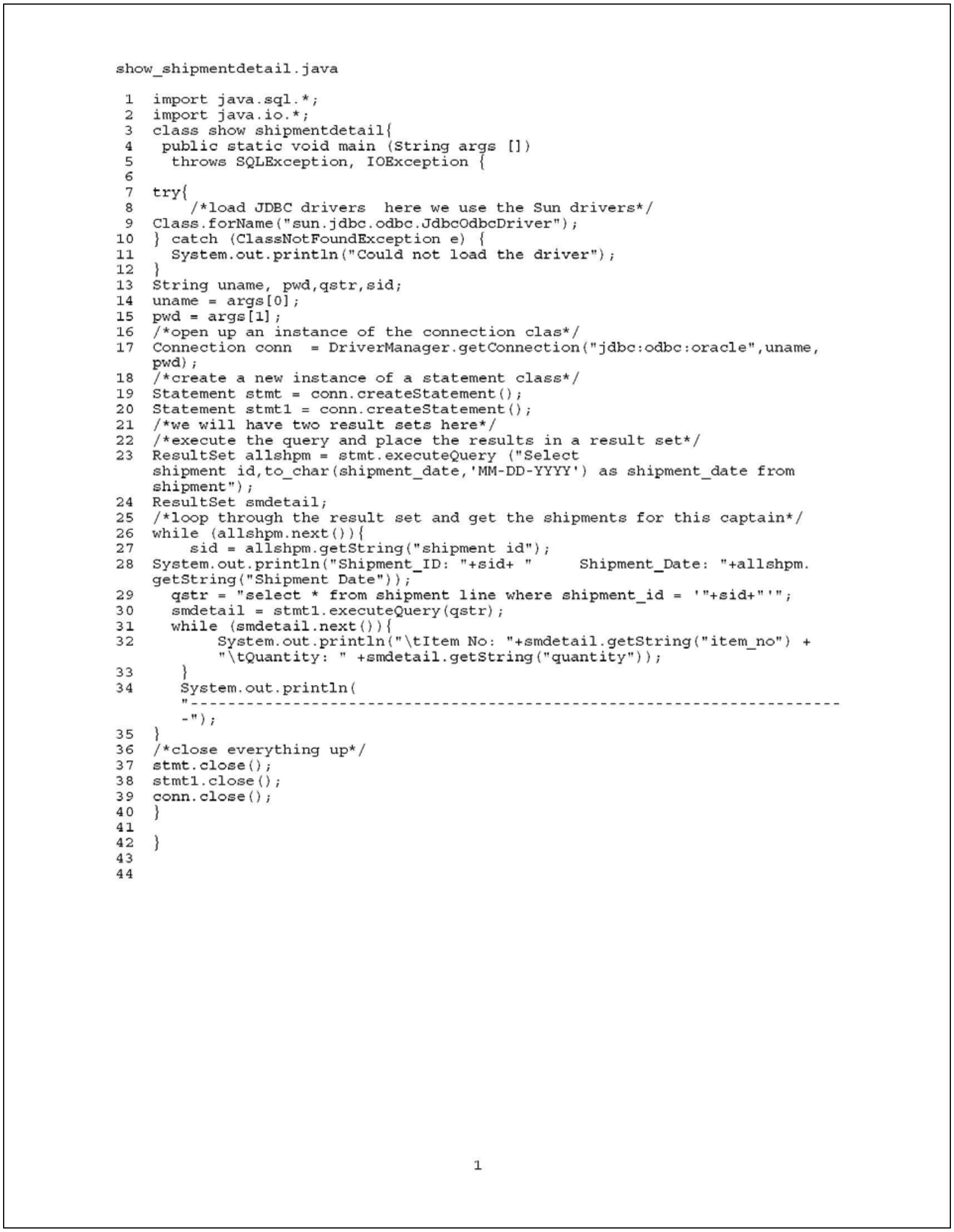
The ResultSet Method and Multiple ResultSets

Table 1 summarizes the methods for the ResultSet class.

Method	**Returns**	**Description**
Next	Boolean	Moves the cursor (or pointer) of resultset to the first row. The cursor is initially positioned before the first row, so you must call next to access the first record. Returns false at end of resultset
Close	Void	Closes resultset
wasNull	Boolean	Returns true if previous column value read was null
getString	String	Retrieves column value of current row in resultset. Can be passed either the column number or column name
getXXX	Depends on type	Same as getString, but for different data types. For example, there is a getFloat, getBoolean and getInt methods. See http://java.sun.com/j2se/1.4.2/docs/api/java/sql/ResultSet.html for a complete list
getMetaData	ResultSetMetaData	Returns metadata on resultset

Table 1

We can open up more than one resultset at a time and work through each one. For example, let's rewrite example 6.1 from Lecture 6 using ODBC. As you may recall, example 6.1 lists all of the line items for each shipment. Figure 11 shows the code.


show_shipmentdetail.java

```
import java.sql.*;
import java.io.*;
class show shipmentdetail{
 public static void main (String args [])
  throws SQLException, IOException {

try{
    /*load JDBC drivers  here we use the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
  System.out.println("Could not load the driver");
}
String uname, pwd,qstr,sid;
uname = args[0];
pwd = args[1];
/*open up an instance of the connection clas*/
Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
pwd);
/*create a new instance of a statement class*/
Statement stmt = conn.createStatement();
Statement stmt1 = conn.createStatement();
/*we will have two result sets here*/
/*execute the query and place the results in a result set*/
ResultSet allshpm = stmt.executeQuery ("Select
shipment id,to_char(shipment_date,'MM-DD-YYYY') as shipment_date from
shipment");
ResultSet smdetail;
/*loop through the result set and get the shipments for this captain*/
while (allshpm.next()){
    sid = allshpm.getString("shipment id");
System.out.println("Shipment_ID: "+sid+ "     Shipment_Date: "+allshpm.
getString("Shipment Date"));
  qstr = "select * from shipment line where shipment_id = '"+sid+"'";
  smdetail = stmt1.executeQuery(qstr);
  while (smdetail.next()){
       System.out.println("\tItem No: "+smdetail.getString("item_no") +
       "\tQuantity: " +smdetail.getString("quantity"));
   }
   System.out.println(
   "--------------------------------------------------------------------
   -");
}
/*close everything up*/
stmt.close();
stmt1.close();
conn.close();
}

}
```

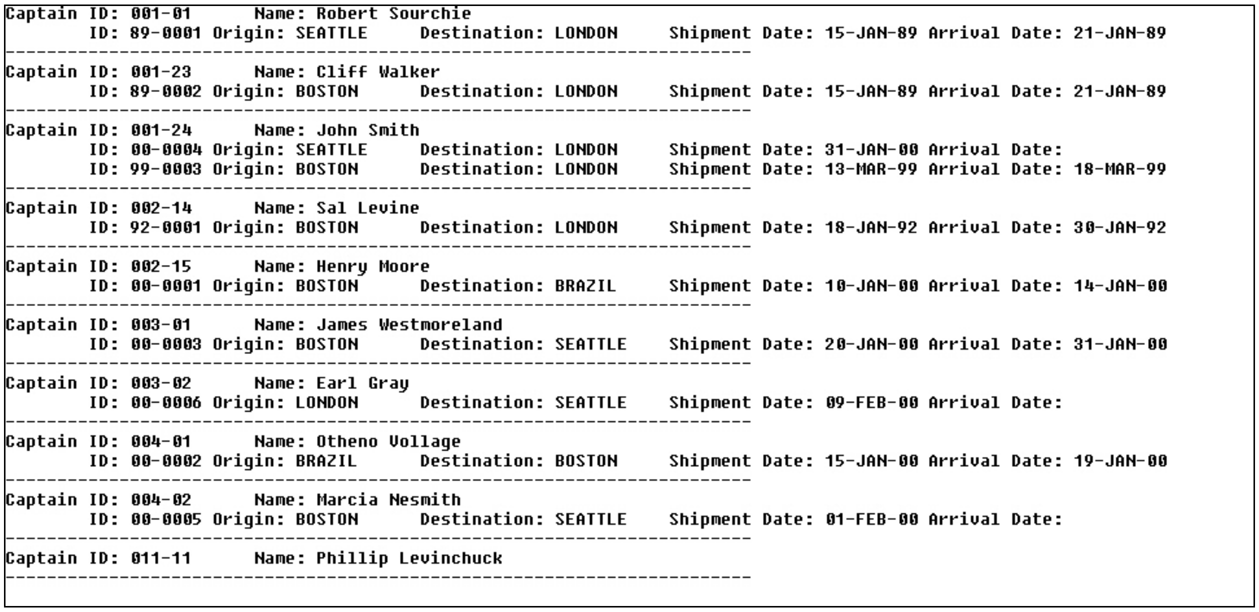
1

Figure 11 Show_shipmentdetail.java

A few things to note here: notice how we declare two instances of Statement (stmt1 and stmt2) Each one will hold a resultset.
An error in your PL/SQL program is called an exception. Exceptions can be caused
Also, notice how we are creating a new resultset each time that we loop through the outer resultset.

<u>On Your Own Exercise</u>

2.1 For each captain, produce a list of each shipment that he has been the captain of, including the shipment_id, origin, destination, date of shipment and date of arrival. The output should look like the following:



```
Captain ID: 001-01      Name: Robert Sourchie
        ID: 89-0001 Origin: SEATTLE     Destination: LONDON     Shipment Date: 15-JAN-89 Arrival Date: 21-JAN-89
------------------------------------------------------------------------
Captain ID: 001-23      Name: Cliff Walker
        ID: 89-0002 Origin: BOSTON      Destination: LONDON     Shipment Date: 15-JAN-89 Arrival Date: 21-JAN-89
------------------------------------------------------------------------
Captain ID: 001-24      Name: John Smith
        ID: 00-0004 Origin: SEATTLE     Destination: LONDON     Shipment Date: 31-JAN-00 Arrival Date:
        ID: 99-0003 Origin: BOSTON      Destination: LONDON     Shipment Date: 13-MAR-99 Arrival Date: 18-MAR-99
------------------------------------------------------------------------
Captain ID: 002-14      Name: Sal Levine
        ID: 92-0001 Origin: BOSTON      Destination: LONDON     Shipment Date: 18-JAN-92 Arrival Date: 30-JAN-92
------------------------------------------------------------------------
Captain ID: 002-15      Name: Henry Moore
        ID: 00-0001 Origin: BOSTON      Destination: BRAZIL     Shipment Date: 10-JAN-00 Arrival Date: 14-JAN-00
------------------------------------------------------------------------
Captain ID: 003-01      Name: James Westmoreland
        ID: 00-0003 Origin: BOSTON      Destination: SEATTLE    Shipment Date: 20-JAN-00 Arrival Date: 31-JAN-00
------------------------------------------------------------------------
Captain ID: 003-02      Name: Earl Gray
        ID: 00-0006 Origin: LONDON      Destination: SEATTLE    Shipment Date: 09-FEB-00 Arrival Date:
------------------------------------------------------------------------
Captain ID: 004-01      Name: Otheno Vollage
        ID: 00-0002 Origin: BRAZIL      Destination: BOSTON     Shipment Date: 15-JAN-00 Arrival Date: 19-JAN-00
------------------------------------------------------------------------
Captain ID: 004-02      Name: Marcia Nesmith
        ID: 00-0005 Origin: BOSTON      Destination: SEATTLE    Shipment Date: 01-FEB-00 Arrival Date:
------------------------------------------------------------------------
Captain ID: 011-11      Name: Phillip Levinchuck
------------------------------------------------------------------------
```

NonQuery SQL Statements

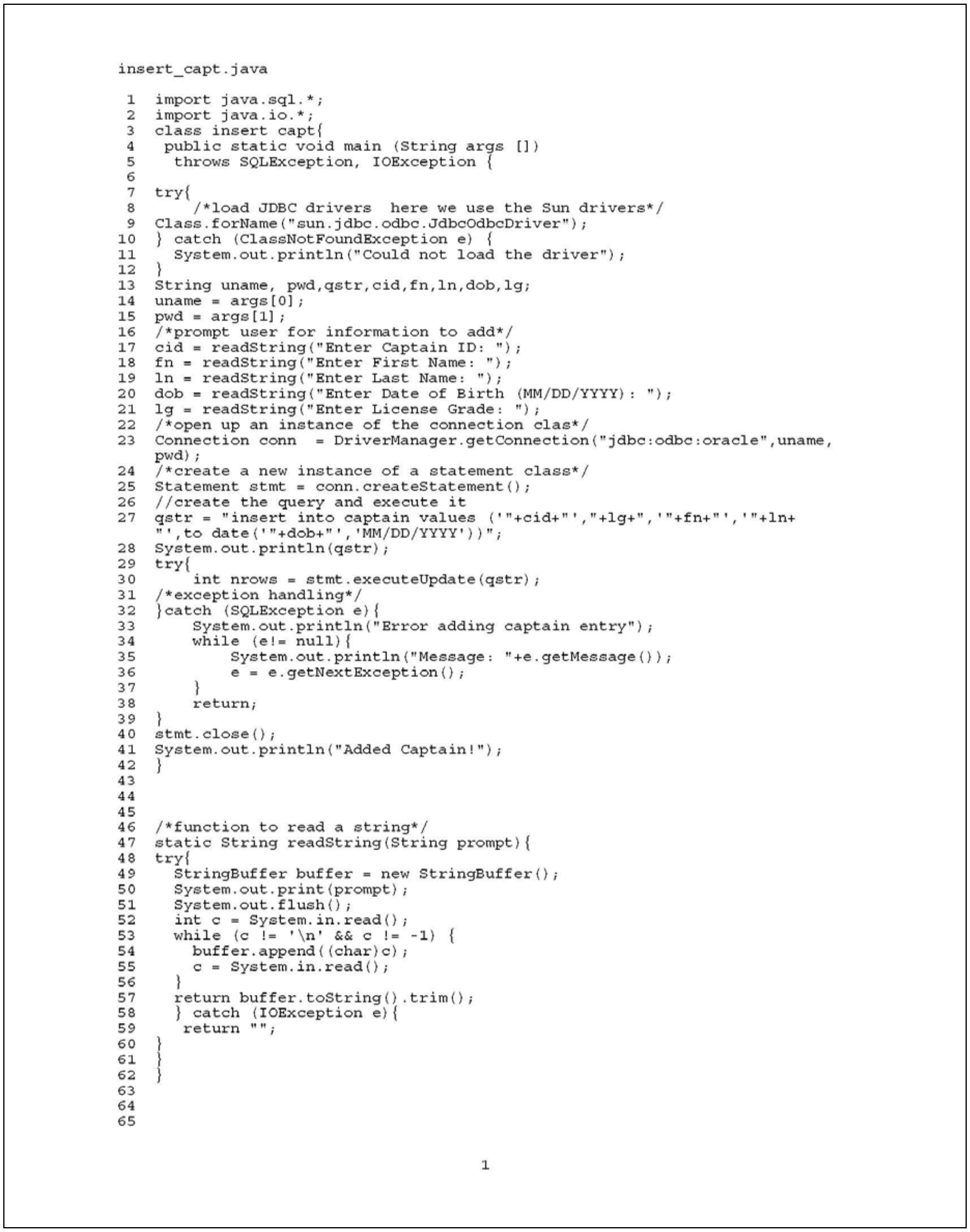
In order to execute nonquery SQL statements (such as DDL command, stored procedures and functions) we need to use one of the following statement object:
Statement
PreparedStatement
CallableStatement

We will take a look at statement and PreparedStatement in this lecture. We will look at CallableStatement next week.

<u>Statement</u>
We have already used Statement in order to execute SQL queries. We can also use the Statement to execute a DDL command, such as an insert, create table or update.

<u>Example 3.1</u>
Figure 12 shows the code for a Java application to prompt the user for information about a captain. This is then added to the captain table. Notice the

use of more exception handling (the try and catch statements). This is needed to be sure that we have successfully completed the operation.



insert_capt.java

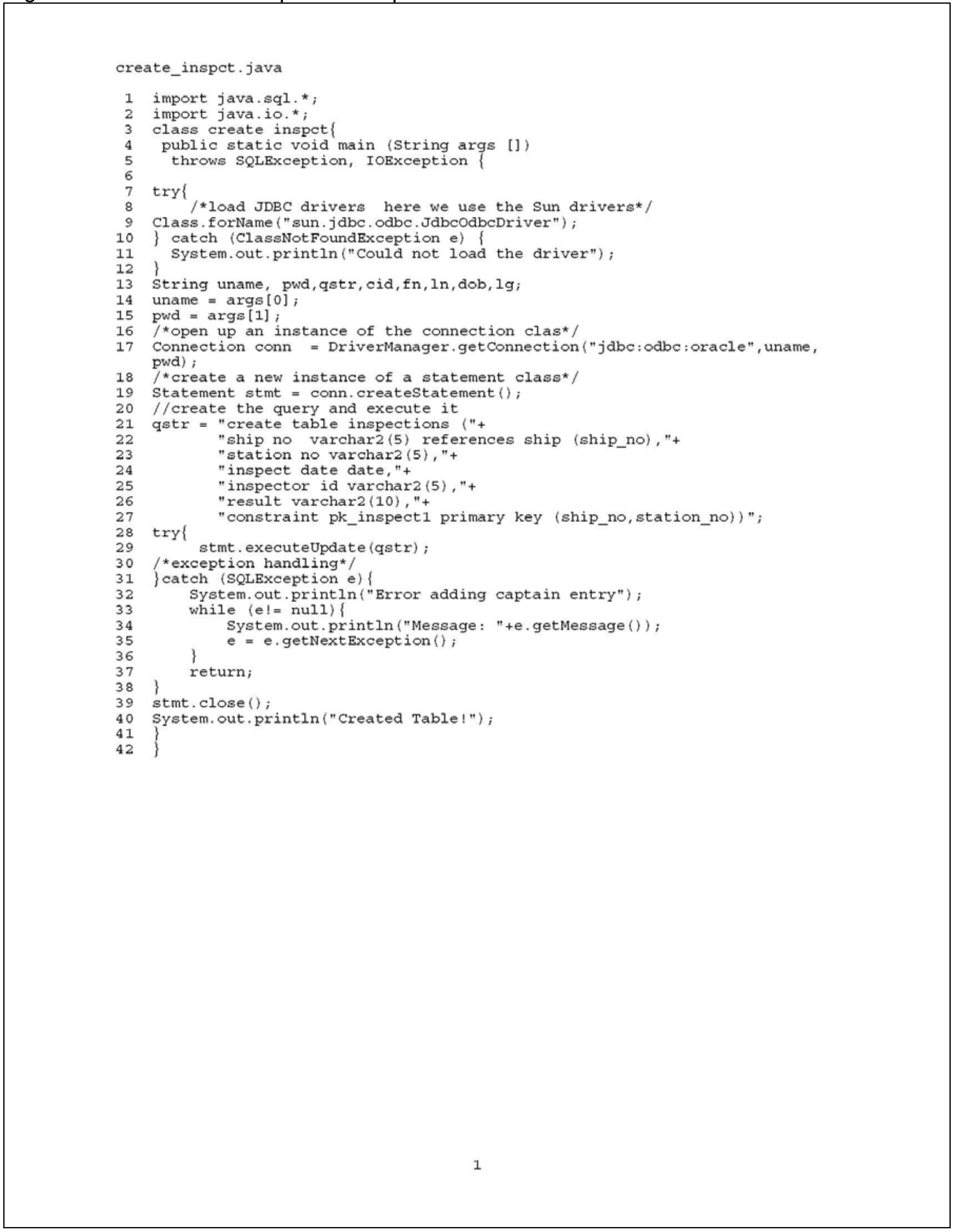
```
 1  import java.sql.*;
 2  import java.io.*;
 3  class insert_capt{
 4   public static void main (String args [])
 5    throws SQLException, IOException {
 6
 7  try{
 8      /*load JDBC drivers  here we use the Sun drivers*/
 9  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
10  } catch (ClassNotFoundException e) {
11    System.out.println("Could not load the driver");
12  }
13  String uname, pwd,qstr,cid,fn,ln,dob,lg;
14  uname = args[0];
15  pwd = args[1];
16  /*prompt user for information to add*/
17  cid = readString("Enter Captain ID: ");
18  fn = readString("Enter First Name: ");
19  ln = readString("Enter Last Name: ");
20  dob = readString("Enter Date of Birth (MM/DD/YYYY): ");
21  lg = readString("Enter License Grade: ");
22  /*open up an instance of the connection clas*/
23  Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
    pwd);
24  /*create a new instance of a statement class*/
25  Statement stmt = conn.createStatement();
26  //create the query and execute it
27  qstr = "insert into captain values ('"+cid+"',"+lg+",'"+fn+"','"+ln+
    "',to_date('"+dob+"','MM/DD/YYYY'))";
28  System.out.println(qstr);
29  try{
30      int nrows = stmt.executeUpdate(qstr);
31  /*exception handling*/
32  }catch (SQLException e){
33      System.out.println("Error adding captain entry");
34      while (e!= null){
35          System.out.println("Message: "+e.getMessage());
36          e = e.getNextException();
37      }
38      return;
39  }
40  stmt.close();
41  System.out.println("Added Captain!");
42  }
43
44
45
46  /*function to read a string*/
47  static String readString(String prompt){
48  try{
49    StringBuffer buffer = new StringBuffer();
50    System.out.print(prompt);
51    System.out.flush();
52    int c = System.in.read();
53    while (c != '\n' && c != -1) {
54      buffer.append((char)c);
55      c = System.in.read();
56    }
57    return buffer.toString().trim();
58    } catch (IOException e){
59     return "";
60  }
61  }
62  }
63
64
65
```

1

Figure 12 insert_capt.java

Notice on line 27 that we are using the to_date function to convert the date from MM/DD/YYYY to an Oracle format.

We can also create tables, delete records and update records using the Statement object. Figure 13 shows an example of a create_table command. Figure 14 shows an example of an update command.

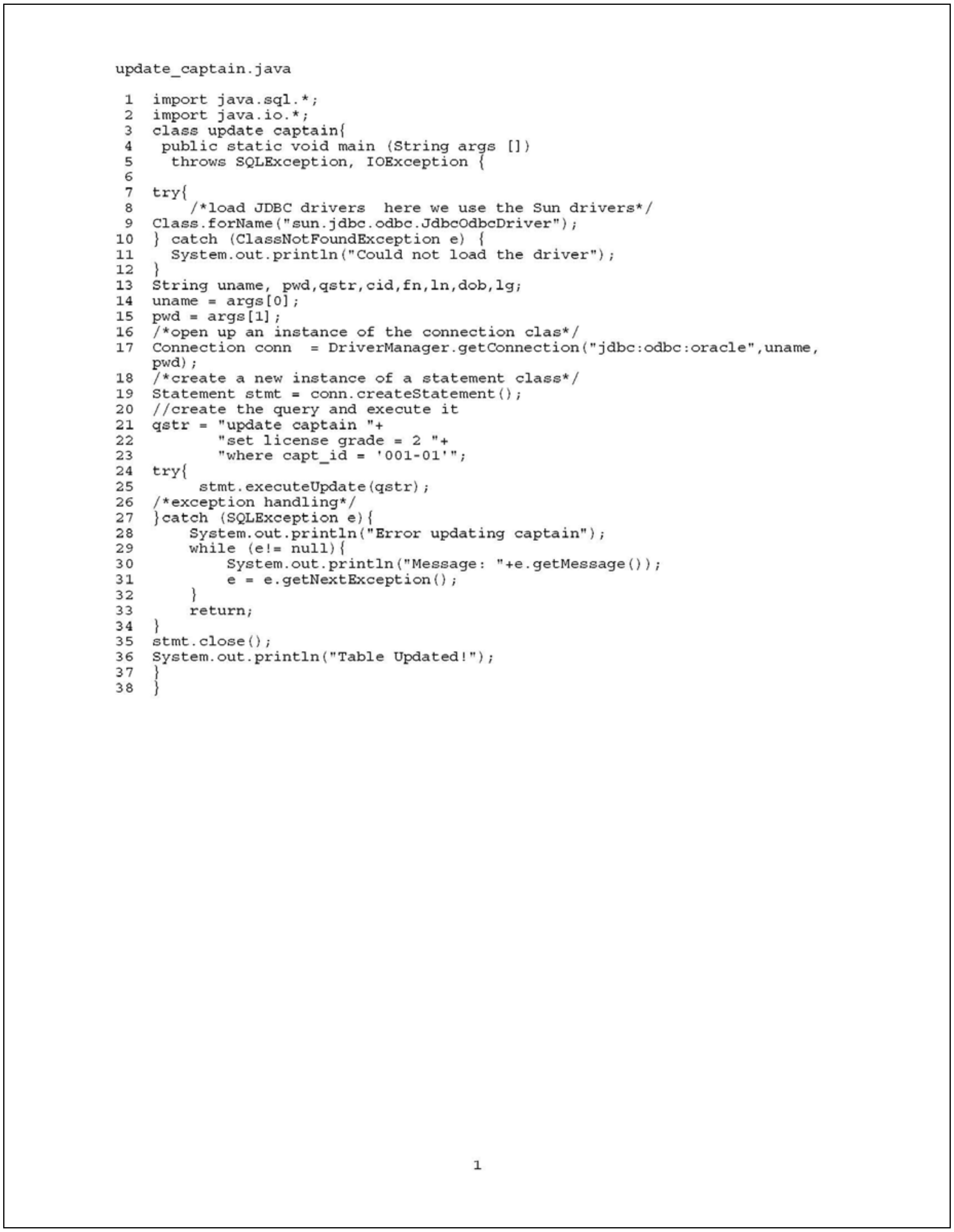


create_inspct.java

```
import java.sql.*;
import java.io.*;
class create inspct{
 public static void main (String args [])
  throws SQLException, IOException {

try{
    /*load JDBC drivers  here we use the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
  System.out.println("Could not load the driver");
}
String uname, pwd,qstr,cid,fn,ln,dob,lg;
uname = args[0];
pwd = args[1];
/*open up an instance of the connection clas*/
Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
pwd);
/*create a new instance of a statement class*/
Statement stmt = conn.createStatement();
//create the query and execute it
qstr = "create table inspections ("+
       "ship no  varchar2(5) references ship (ship_no),"+
       "station no varchar2(5),"+
       "inspect date date,"+
       "inspector id varchar2(5),"+
       "result varchar2(10),"+
       "constraint pk_inspect1 primary key (ship_no,station_no))";
try{
     stmt.executeUpdate(qstr);
/*exception handling*/
}catch (SQLException e){
    System.out.println("Error adding captain entry");
    while (e!= null){
        System.out.println("Message: "+e.getMessage());
        e = e.getNextException();
    }
    return;
}
stmt.close();
System.out.println("Created Table!");
}
}
```

1

Figure 13 create_inspct.java



update_captain.java

```
import java.sql.*;
import java.io.*;
class update captain{
 public static void main (String args [])
  throws SQLException, IOException {

try{
    /*load JDBC drivers  here we use the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
  System.out.println("Could not load the driver");
}
String uname, pwd,qstr,cid,fn,ln,dob,lg;
uname = args[0];
pwd = args[1];
/*open up an instance of the connection clas*/
Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
pwd);
/*create a new instance of a statement class*/
Statement stmt = conn.createStatement();
//create the query and execute it
qstr = "update captain "+
       "set license grade = 2 "+
       "where capt_id = '001-01'";
try{
     stmt.executeUpdate(qstr);
/*exception handling*/
}catch (SQLException e){
    System.out.println("Error updating captain");
    while (e!= null){
        System.out.println("Message: "+e.getMessage());
        e = e.getNextException();
    }
    return;
}
stmt.close();
System.out.println("Table Updated!");
}
}
```

1

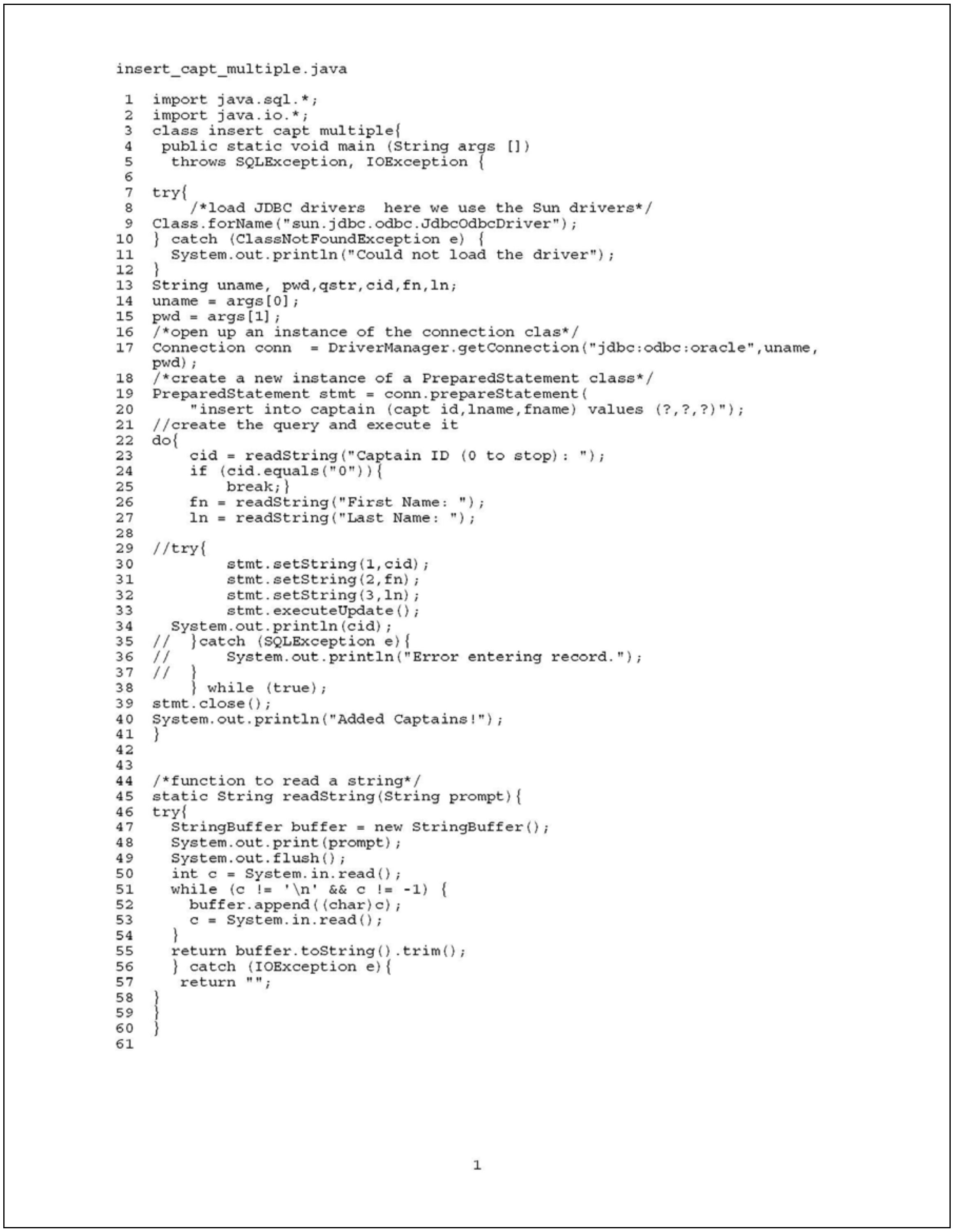
Figure 14 Update_Capt.java

PreparedStatement Object

The PreparedStatement object is used to execute a statement several times with different parameters. For example, you could insert a series of captains. In this lecture, we will take a look at a simple example. Next week, we will look at how to read from a file and insert into a table.

Example 3.2

In our example, we wish to insert a number of captains into the captain table. Figure 15 shows the code for this. Notice that we first define the prepared statement with some parameters. We later specify what values those parameters take and then execute the query.



insert_capt_multiple.java

```
import java.sql.*;
import java.io.*;
class insert capt multiple{
 public static void main (String args [])
  throws SQLException, IOException {

try{
    /*load JDBC drivers  here we use the Sun drivers*/
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
  System.out.println("Could not load the driver");
}
String uname, pwd,qstr,cid,fn,ln;
uname = args[0];
pwd = args[1];
/*open up an instance of the connection clas*/
Connection conn  = DriverManager.getConnection("jdbc:odbc:oracle",uname,
pwd);
/*create a new instance of a PreparedStatement class*/
PreparedStatement stmt = conn.prepareStatement(
    "insert into captain (capt id,lname,fname) values (?,?,?)");
//create the query and execute it
do{
    cid = readString("Captain ID (0 to stop): ");
    if (cid.equals("0")){
        break;}
    fn = readString("First Name: ");
    ln = readString("Last Name: ");

//try{
        stmt.setString(1,cid);
        stmt.setString(2,fn);
        stmt.setString(3,ln);
        stmt.executeUpdate();
  System.out.println(cid);
//  }catch (SQLException e){
//      System.out.println("Error entering record.");
//  }
    } while (true);
stmt.close();
System.out.println("Added Captains!");
}


/*function to read a string*/
static String readString(String prompt){
try{
  StringBuffer buffer = new StringBuffer();
  System.out.print(prompt);
  System.out.flush();
  int c = System.in.read();
  while (c != '\n' && c != -1) {
    buffer.append((char)c);
    c = System.in.read();
  }
  return buffer.toString().trim();
  } catch (IOException e){
   return "";
}
}
}
```

1

<u>On Your Own Exercise</u>

3.1 Write a program in Java using JDBC to create a table which has the capt_id, an origin and destination pair as well as a count of the number of shipments shipped from that origin and to that destination with that captain. Once the table has been created, your program should insert rows for all captains in the captain table.

Summary

In this lecture, we covered the basics of JDBC including the Statement and PreparedStatement classes. Next, we will continue with a discussion of the CallableStatement class as well as using JDBC within applets.