```java
/**
 * --------------------------------------------------------------------------
 * File name: labTwelve.java
 * Project name: Lab 12
 * --------------------------------------------------------------------------
 * Creator's name and email: Jordan Hall (halljb1@etsu.edu)
 *                           from a solution written by Donald Sanderson
 * Course-Section:  CSCI2020-ALL
 * Creation Date: Nov 30, 2011
 * Date of Last Modification: Dec 2, 2011
 * --------------------------------------------------------------------------
 */

// Step 1 - get all necessary imports
import java.sql.Date;           // Class for dates retrieved from a DB
import java.sql.DriverManager;  // DriverManager for establishing a DB link
import java.sql.SQLException;   // SQLException exceptions for DB classes
import java.sql.Connection;     // Connection establishes a logon to the DB
import java.sql.Statement;      // Holds and executes the sql statements
import java.sql.ResultSet;      // Holds the results of a query

// Note that this block of imports is a subset of ...
//
//import java.sql.*;
//
// ...which imports all of these classes, along with some others that we don't
//      use in this exercise.  Depending on what problems your solution is meant
//      to solve, you may want to import more or less, and in some cases may want
//      to just import the entire package with the commented import statement.
//      A description of the java.sql package is available at
//  http://docs.oracle.com/javase/1.4.2/docs/api/java/sql/package-summary.html

public class labTwelve {
    /* This class implements the solution to lab 12, including handling NULL
     * values in the GIVEN_NAME column.
     */

    public static void main(String[] args)
    {
        // Step 2 - declare ALL necessary variables

        // First, we need strings for building our connection and statement(s)
        String connection;      // holds the connection string
        String username;        // holds the user name
        String password;        // holds the password
        String sqlStatement;    // holds the SQL statement to be run

        // JDBC variables will use those strings
        Connection myConn;      // JDBC Connection object
        Statement myState;      // JDBC Statement object
        ResultSet myResults;    // JDBC ResultSet object


        // Now we need variables to hold values in our while loop
```

```
        String givenName;       // holds the given name for the current row
        String familyName;      // holds the family name for the current row
        int salary = 0;         // holds the salary for the current row
        int oldSalary = 0;      // holds the salary from the previous row
        String compare;         // holds "Higher", "Less" or "The Same"
        // end declare variables


        // Step 3 - start a try block because you may have errors
        try
        {
            // provide some feedback so we know what's happening
            System.out.println("Begin Connection Process...");

            // Step 4 - link our class to the JDBC driver
            System.out.println("Linking class to JDBC driver...");
            DriverManager.registerDriver(new oracle.jdbc.OracleDriver());

            // Step 5 - establish a connection to the database and register with
            //  the driver
            // first, build these strings up to use with the Connection object
            connection = "jdbc:oracle:thin:@pythia.etsu.edu:1521:csdb";
            username = "heyitsjordan";
            password = "lookitshispassword!";
            // then, build your connection/register with the driver
            myConn = DriverManager.getConnection(connection, username, password);
            // now we have a connection

            // Step 6 - create a Statement object
            // we are going to build up the object declared earlier,
            //     and register it with our connection
            myState = myConn.createStatement();

            // Step 7 - create a ResultSet object
            // we declared the object already, in accordance with coding standards,
            //     so now we can just set it to null and use it in a bit (we
            //     could have done that when we declared the variable, but here
            //     we are anyway)
            myResults = null;

            // Step 8 - build and execute a query
            // we will use the sqlStatement string we declared earlier to hold a
            //      query that we will build up a piece at a time, then we will
            //      have the Statement object execute the query, sending the
            //      output into the ResultSet variable we created.
            sqlStatement = "select INSTRUCTOR.GIVEN_NAME, INSTRUCTOR.FAMILY_NAME,
INSTRUCTOR.SALARY";
            sqlStatement += " from INSTRUCTOR";
            sqlStatement += " order by INSTRUCTOR.FAMILY_NAME";

            // we can look at the concatenated query to find any errors, if we like
            System.out.println("Query to be executed: " + sqlStatement);

            // and now to execute and catch the output
```

```
        myResults = myState.executeQuery(sqlStatement);


        // Step 9 - process one row at a time
        // now that the query has been run, we need to look through the
        //      results and do our stuff to them.  We will set up a while
        //      loop to look through the results one at a time as long as
        //      there are results to look through.  We can use the next()
        //      method in the ResultSet class to see if there is another
        //      row to retrieve.
        while(myResults.next())
        {
            // we can get the individual fields from the results by using
            //      the column name and making sure we are using a compatible
            //      variable to hold the result

            // INSTRUCTOR.FAMILY_NAME is a string, we will use the familyName
            //      String that we declared earlier to hold it
            familyName = myResults.getString("FAMILY_NAME");

            // INSTRUCTOR.GIVEN_NAME is a string as well, same deal as before
            givenName = myResults.getString("GIVEN_NAME");
            // and now we can use a simple if statement to see if the result
            //      was null.
            //      note that we are not comparing the result to a string
            //      with a value "NULL", we are seeing if the field for this
            //      row was a null value. if it does, we will just substitute
            //      a few blank spaces for the name
            if (myResults.wasNull())
            {
                givenName = "       ";
            }

            // INSTRUCTOR.SALARY can be held by an int
            salary = myResults.getInt("SALARY");
            // and now we can compare the current salary to the previous
            //      salary to see if it is higher, lower or the same
            if (salary > oldSalary)
            {
                compare = "Higher";
            }
            else if (salary < oldSalary)
            {
                compare = "Lower";
            }
            else
            {
                compare = "The Same";
            }

            // Now that we have all of our pieces, we can put them together
            //      in a nice, pretty output string
            System.out.println("Instructor: " + givenName + " " +
                    familyName + "'s salary of " + salary + " is " +
```

```
                        compare + " than the previous instructor's salary of " +
                                oldSalary + ".");

                // and finally, we need to change the value of the oldSalary to
                //      our current salary
                oldSalary = salary;
            } // end while

            // Step 9 - close stuff in the opposite order we opened them
            myResults.close();  // first, the ResultSet
            myState.close();    // then the Statement
            myConn.close();     // and finally the Connection

        } // end try
        catch (SQLException mySqlError)
        {
            // really simple error handling.  We are just going to catch the
            //      error and dump the error stack
            System.out.println("An Error has occured, dumping error stack...");
            mySqlError.printStackTrace();
        } // end catch


    } // end main

}// end class labTwelve
```