CSCI 2020 Database Fundamentals
HW8

Submit the homework as **JAVA documents** called
HW8A.java *and* ***HW8B.java*** to the drop box.

TASKS

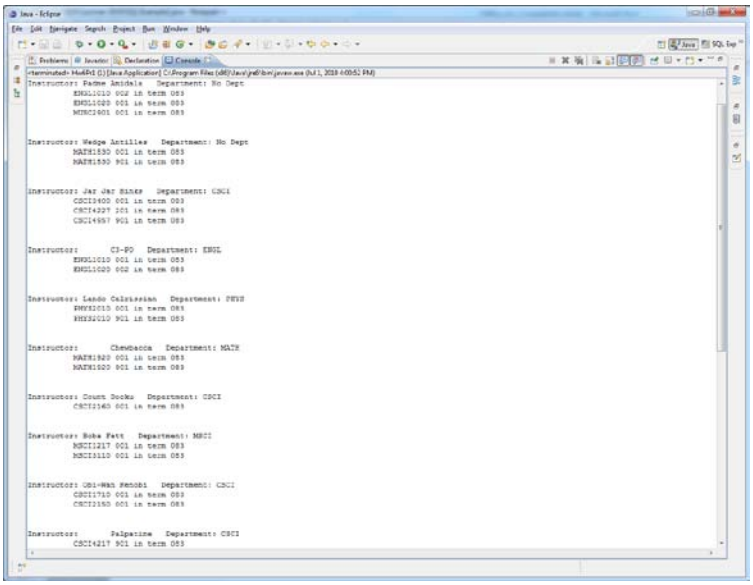
You are to write a java program called **HW8A.java** which will **update the instructors' salaries** by giving raises based on the following scale:

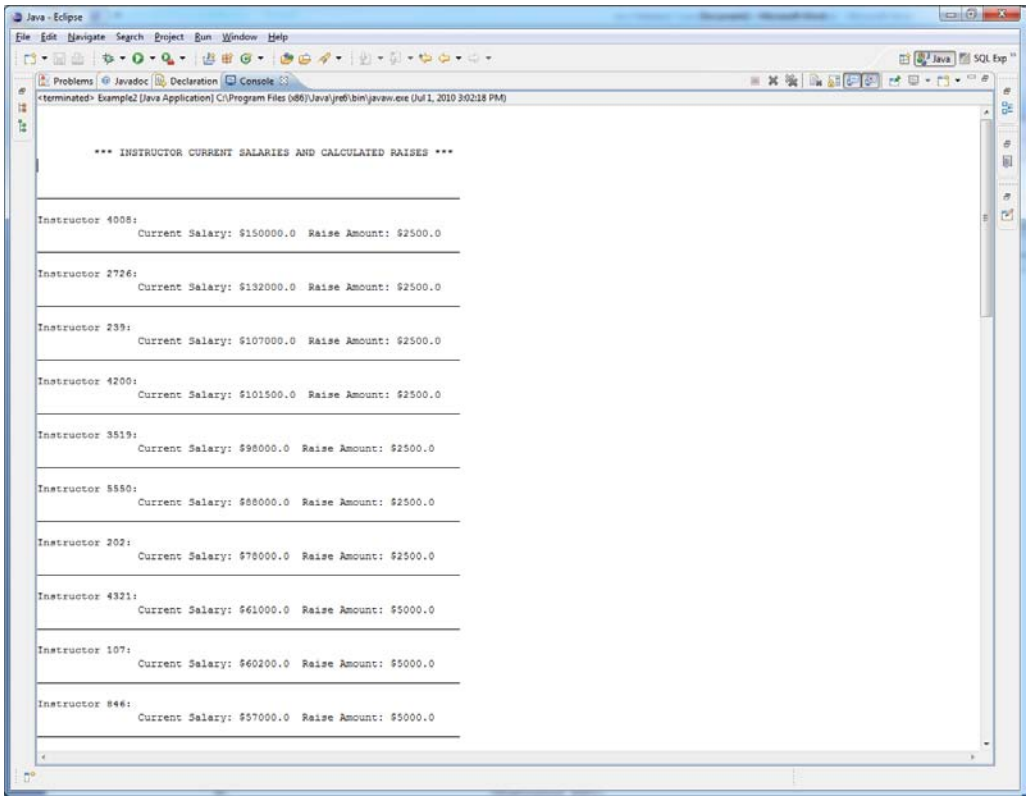
- If the instructor is making less than $50,000 a $10,000 raise will be given
- If the instructor is making more than $50,000 but less than $75,000 a $5,000 raise will be given
- If the instructor is making $75,000 or more a $2,500 raise will be given

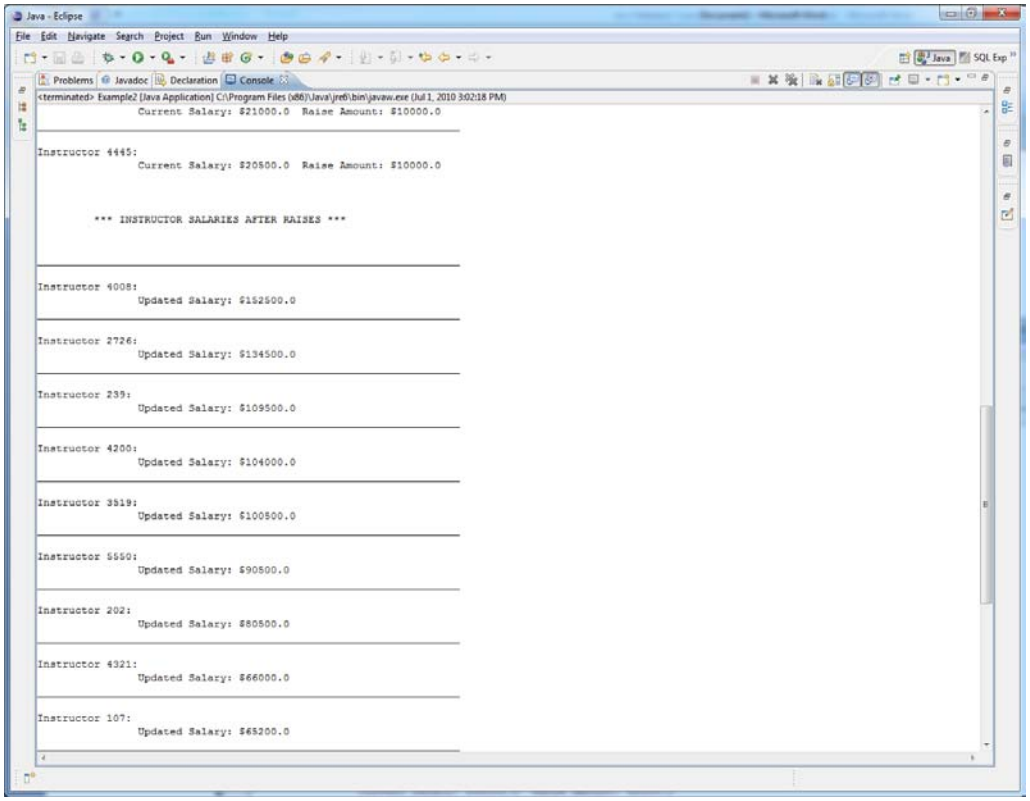
You are to display the current salaries and the amount of raise to be given for each instructor in descending order by salary. Once this information has been displayed for each instructor, you will query the database and display all instructors with their updated salaries in descending order by salary.

The output should be in the format shown in the screen shots *on the following page*.

You are to write a java program called **HW8B.java** which will **print the teaching load** for instructors during the Fall 08 (083) semester. The output should be in the format shown in the screen shot *below*. (Be sure to add a rollback at the end so it does not permanently record the changes. This will help when I'm grading.)




Java - Eclipse
File Edit Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> Example2 [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Jul 1, 2010 3:02:18 PM)
*** INSTRUCTOR CURRENT SALARIES AND CALCULATED RAISES ***
Instructor 4008:
Current Salary: $150000.0 Raise Amount: $2500.0
Instructor 2726:
Current Salary: $132000.0 Raise Amount: $2500.0
Instructor 239:
Current Salary: $107000.0 Raise Amount: $2500.0
Instructor 4200:
Current Salary: $101500.0 Raise Amount: $2500.0
Instructor 3519:
Current Salary: $98000.0 Raise Amount: $2500.0
Instructor 5550:
Current Salary: $88000.0 Raise Amount: $2500.0
Instructor 202:
Current Salary: $78000.0 Raise Amount: $2500.0
Instructor 4321:
Current Salary: $61000.0 Raise Amount: $5000.0
Instructor 107:
Current Salary: $60200.0 Raise Amount: $5000.0
Instructor 846:
Current Salary: $57000.0 Raise Amount: $5000.0


Java - Eclipse
File Edit Navigate Search Project Run Window Help
Problems Javadoc Declaration Console
<terminated> Example2 [Java Application] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (Jul 1, 2010 3:02:18 PM)
Current Salary: $21000.0 Raise Amount: $10000.0
Instructor 4445:
Current Salary: $20500.0 Raise Amount: $10000.0
*** INSTRUCTOR SALARIES AFTER RAISES ***
Instructor 4008:
Updated Salary: $152500.0
Instructor 2726:
Updated Salary: $134500.0
Instructor 239:
Updated Salary: $109500.0
Instructor 4200:
Updated Salary: $104000.0
Instructor 3519:
Updated Salary: $100500.0
Instructor 5550:
Updated Salary: $90500.0
Instructor 202:
Updated Salary: $80500.0
Instructor 4321:
Updated Salary: $66000.0
Instructor 107:
Updated Salary: $65200.0

```
/* HW8A.java
-----------------------------------------------------------------------
Programmer's Name: Brandon Buchanan
Course: CSCI2020                          Section Number: 002
Creation Date: 12/6/2011        Date of Last Modification: 12/7/2011
E-mail address: zbwb8@goldmail.etsu.edu
-----------------------------------------------------------------------
Purpose -

 SQL integration with Java programs
-----------------------------------------------------------------------
Notes on specifications, special algorithms, and assumptions:
Designed to work on the database created by the schooldb.sql
script
*/

import java.sql.DriverManager;  // DriverManager for establishing linkage to DB
import java.sql.SQLException;   // SQLException handles exception 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

public class HW8A {

    // Global fields
    private static String myConnStr;    // holds the connection string
    private static String myUser;       // holds the user name
    private static String myPasswd;     // holds the user password
    private static Connection myCon;    // holds sql connection object

    /**
     * Main method
     * @param args
     */
    public static void main(String[] args)
    {
        // Connect to the SQL server
        openConnection();

        String strQuery = "";
        strQuery = "select INSTRUCTOR.ID, INSTRUCTOR.SALARY";
        strQuery += " from INSTRUCTOR";
        strQuery += " order by INSTRUCTOR.SALARY desc";

        // Display retreived information to the console screen
        ResultSet results = executeQuery(strQuery);
        displayCurrentSalaryResults(results);

        // Update the instructors' salaries
        ResultSet results2 = executeQuery(strQuery);
        updateSalaries(results2);

        // Display updated salary results
        ResultSet results3 = executeQuery(strQuery);
```

```
        displayUpdatedSalaryResults(results3);

        // Roll back changes
        System.out.println("\n\nRolling back changes...");
        executeQuery("rollback");

        // Close the SQL connection
        closeConnection();
    }

    /***
     * Update the salaries of the instructors
     */
    private static void updateSalaries(ResultSet results)
    {
        try
        {
            // Loop through the result set
            while (results.next())
            {
                // Variables for returned elements
                String strID = results.getString("ID");
                int salary = results.getInt("SALARY");
                int raiseAmount = 0;

                // Determine raise amount
                if(salary < 50000)
                    raiseAmount = 10000;
                else if(salary>=50000 && salary<75000)
                    raiseAmount=5000;
                else
                    raiseAmount=2500;

                // Construct SQL update string
                String strUpdate = "";
                strUpdate += "update INSTRUCTOR";
                strUpdate += " set INSTRUCTOR.SALARY = " + salary+raiseAmount;
                strUpdate += " where INSTRUCTOR.ID = " + strID;

                // Execute the update query
                executeUpdate(strUpdate);
            }
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to update the salary.");
        }
    }

    /**
     * Displays the results of the INSTRUCTOR query.
     * @param results
     */
    private static void displayCurrentSalaryResults(ResultSet results)
```

```
    {
        System.out.println("\n\n   *** INSTRUCTOR CURRENT SALARIES AND CALCULATED RAISES
        ***\n\n");

        try
        {
            // Loop through the result set
            while (results.next())
            {
                // Variables for returned elements
                String strID = results.getString("ID");
                int salary = results.getInt("SALARY");
                int raiseAmount = 0;

                // Determine raise amount
                if(salary < 50000)
                    raiseAmount = 10000;
                else if(salary>=50000 && salary<75000)
                    raiseAmount=5000;
                else
                    raiseAmount=2500;

                // Construct displayed string
                String output = "";
                output += "Instructor: " + strID;
                output += "\n\tCurrent salary: " + salary;
                output += "\tRaise amount: " + raiseAmount;

                // Display the output to the console
                System.out.println(output);
            }
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to retrieve a result.");
        }
    }

    /***
     * Updates the salaries of the instructors
     * @param results SQL result of the INSTRUCTOR table for ID and SALARY
     */
    private static void displayUpdatedSalaryResults(ResultSet results)
    {
        System.out.println("\n\n   *** INSTRUCTOR SALARIES AFTER RAISES ***\n\n");

        try
        {
            // Loop through the result set
            while (results.next())
            {
                // Variables for returned elements
                String strID = results.getString("ID");
                int salary = results.getInt("SALARY");
```

```
            int raiseAmount = 0;

            // Determine raise amount
            if(salary < 50000)
                raiseAmount = 10000;
            else if(salary>=50000 && salary<75000)
                raiseAmount=5000;
            else
                raiseAmount=2500;

            // Construct displayed string
            String output = "";
            output += "Instructor: " + strID;
            output += "\n\tUpdated salary: " + salary;

            // Display the output to the console
            System.out.println(output);
        }
    }
    catch(SQLException sqle)
    {
        System.out.println("Unable to retrieve a result.");
    }
}

/**
 * Retrieves the result of a given SQL query.
 * @return
 */
private static ResultSet executeQuery(String query)
{
    Statement myQueryStmt = null;

    try
    {
        myQueryStmt = myCon.createStatement();
    }
    catch(SQLException sqle)
    {
        System.out.println("Unable to create SQL statement.");
    }
    catch(NullPointerException npe)
    {
        System.out.println("Unable to create SQL statement because the statement was null.");
    }

    ResultSet myResults = null;
    try
    {
        myResults = myQueryStmt.executeQuery(query);
    }
    catch(SQLException sqle)
    {
        System.out.println("Unable to retrieve the results of the query.");
```

```
        }

        return myResults;
    }

    /**
     * Retrieves the result of a given SQL query.
     * @return
     */
    private static void executeUpdate(String query)
    {
        Statement myQueryStmt = null;

        try
        {
            myQueryStmt = myCon.createStatement();
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to create SQL statement.");
        }
        catch(NullPointerException npe)
        {
            System.out.println("Unable to create SQL statement because the statement was null.");
        }

        int rowsReturned = -1;
        try
        {
            rowsReturned = myQueryStmt.executeUpdate(query);
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to execute update query - the number of rows returned
            was " + rowsReturned);
        }

        return;
    }

    /**
     * Connects to ETSU's Pythia SQL server.
     */
    private static void openConnection()
    {
        System.out.println("Connecting to the SQL server...");

        try
        {
            DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to register the driver.");
```

```
        }

        // Server connection information and user credentials
        myConnStr = "jdbc:oracle:thin:@pythia.etsu.edu:1521:csdb";
        myUser = "zbwb8";
        myPasswd = "Ghd4fvw437FD";

        myCon = login(myConnStr,myUser,myPasswd);  //establish connection

        System.out.println("SQL connection successful.");
    }

    /**
     * Logs into ETSU's Pythia SQL server.
     *
     * @param server server path to connect to
     * @param userName user name to be used
     * @param password password for the user account
     * @return
     */
    private static Connection login(String server, String userName, String password)
    {
        Connection sqlConn = null;

        try
        {
            sqlConn = DriverManager.getConnection(server,userName,password);
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to establish the SQL connection.");
        }

        return sqlConn;
    }

    /**
     * Closes the SQL connection.
     */
    private static void closeConnection()
    {
        System.out.println("Closing the SQL connection...");

        try
        {
            myCon.close();
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to close the SQL connection.");
        }

        System.out.println("SQL connection closed.");
    }
```

```
}
```

```
/* HW8B.java
-----------------------------------------------------------------------
Programmer's Name: Brandon Buchanan
Course: CSCI2020                          Section Number: 002
Creation Date: 12/6/2011        Date of Last Modification: 12/7/2011
E-mail address: zbwb8@goldmail.etsu.edu
-----------------------------------------------------------------------
Purpose -

 SQL integration with Java programs
-----------------------------------------------------------------------
Notes on specifications, special algorithms, and assumptions:
Designed to work on the database created by the schooldb.sql
script
*/

import java.sql.DriverManager;  // DriverManager for establishing linkage to DB
import java.sql.SQLException;   // SQLException handles exception 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

public class HW8B {

    // Global fields
    private static String myConnStr;    // holds the connection string
    private static String myUser;       // holds the user name
    private static String myPasswd;     // holds the user password
    private static Connection myCon;    // holds sql connection object

    /**
     * Main method
     * @param args
     */
    public static void main(String[] args)
    {
        // Connect to the SQL server
        openConnection();

        String strQuery = "";
        strQuery = "select INSTRUCTOR.GIVEN_NAME, INSTRUCTOR.FAMILY_NAME, " +
                   "INSTRUCTOR.DEPARTMENT, SECTION.COURSE, SECTION.CODE";
        strQuery += " from INSTRUCTOR";
        strQuery += " join SECTION on INSTRUCTOR.ID = SECTION.INSTRUCTOR";
        strQuery += " where SECTION.TERM = '083'";

        // Display retreived information to the console screen
        ResultSet results = executeQuery(strQuery);
        displayResults(results);

        // Roll back changes
        System.out.println("\n\nRolling back changes...");
        executeQuery("rollback");
```

```
        // Close the SQL connection
        closeConnection();
    }

    /**
     * Displays the results of the INSTRUCTOR query.
     * @param results
     */
    private static void displayResults(ResultSet results)
    {
        System.out.println("\n\n   *** INSTRUCTOR COURSE LOAD FOR TERM 083 ***\n");

        try
        {
            String previousFirstName="test";
            String previousLastName="test";

            // Loop through the result set
            while (results.next())
            {
                // Variables for returned elements
                String firstName = results.getString("GIVEN_NAME");
                String lastName = results.getString("FAMILY_NAME");
                String department = results.getString("DEPARTMENT");
                String course = results.getString("COURSE");
                String code = results.getString("CODE");

                // If a first name doesn't exist, then clear the variable
                if(firstName==null)
                    firstName="";

                if(department==null)
                    department="No Department";

                String output="";
                if(firstName.equals(previousFirstName) && lastName.equals(previousLastName))
                {
                    // Construct displayed string
                    output = "\t" + course + " " + code + " in term 083";
                }
                else
                {
                    // Construct new instructor displayed string
                    output = "\nInstructor: " + firstName + " " + lastName;
                    output += "\tDepartment: " + department;
                }

                previousFirstName=firstName;
                previousLastName=lastName;

                // Display the output to the console
                System.out.println(output);
            }
        }
```

```
        catch(SQLException sqle)
        {
            System.out.println("Unable to retrieve a result.");
        }
    }

    /**
     * Retrieves the result of a given SQL query.
     * @return
     */
    private static ResultSet executeQuery(String query)
    {
        Statement myQueryStmt = null;

        try
        {
            myQueryStmt = myCon.createStatement();
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to create SQL statement.");
        }
        catch(NullPointerException npe)
        {
            System.out.println("Unable to create SQL statement because the statement was null.");
        }

        ResultSet myResults = null;
        try
        {
            myResults = myQueryStmt.executeQuery(query);
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to retrieve the results of the query.");
        }

        return myResults;
    }

    /**
     * Retrieves the result of a given SQL query.
     * @return
     */
    private static void executeUpdate(String query)
    {
        Statement myQueryStmt = null;

        try
        {
            myQueryStmt = myCon.createStatement();
        }
        catch(SQLException sqle)
        {
```

```
            System.out.println("Unable to create SQL statement.");
        }
        catch(NullPointerException npe)
        {
            System.out.println("Unable to create SQL statement because the statement was null.");
        }

        int rowsReturned = -1;
        try
        {
            rowsReturned = myQueryStmt.executeUpdate(query);
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to execute update query - the number of rows returned
            was " + rowsReturned);
        }

        return;
    }

    /**
     * Connects to ETSU's Pythia SQL server.
     */
    private static void openConnection()
    {
        System.out.println("Connecting to the SQL server...");

        try
        {
            DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to register the driver.");
        }

        // Server connection information and user credentials
        myConnStr = "jdbc:oracle:thin:@pythia.etsu.edu:1521:csdb";
        myUser = "zbwb8";
        myPasswd = "Ghd4fvw437FD";

        myCon = login(myConnStr,myUser,myPasswd);  //establish connection

        System.out.println("SQL connection successful.");
    }

    /**
     * Logs into ETSU's Pythia SQL server.
     *
     * @param server server path to connect to
     * @param userName user name to be used
     * @param password password for the user account
     * @return
```

```
     */
    private static Connection login(String server, String userName, String password)
    {
        Connection sqlConn = null;

        try
        {
            sqlConn = DriverManager.getConnection(server,userName,password);
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to establish the SQL connection.");
        }

        return sqlConn;
    }

    /**
     * Closes the SQL connection.
     */
    private static void closeConnection()
    {
        System.out.println("Closing the SQL connection...");

        try
        {
            myCon.close();
        }
        catch(SQLException sqle)
        {
            System.out.println("Unable to close the SQL connection.");
        }

        System.out.println("SQL connection closed.");
    }
}
```