TCSS142 Introduction to Object-Oriented Programming
Programming Assignment 4

DUE: July 21, 2013 by 12 midnight

With a Mid-Term Exam arriving next week on Wednesday, the complexity of this assignment is much less than previous ones. However, it does require an understanding of most concepts learned prior to chapter 4 with the addition of both sequential and nested selection (if;if;if; & if/else statements).

You are to write an interactive program that receives employee time worked for a current week, pay rate information, and name and then echoes this information along with the weekly gross pay to the console. This process is repeated based on user input of how many employees to process.

Once all employees are processed, a congratulatory message should be displayed based on the employee with most hours worked. One of four specific messages will be displayed for this ambitious employee based upon how many hours he or she worked:

More than 56 hours: WOW!!!! What a Dynamo! Bruce Willis Worked 98.50 Hours this WEEK!
More than 48 hours: Bruce Wayne is Such a WorkHorse! Looks Like You Worked 52.00 Hours this WEEK!
More than 40 hours: Well, Good For YOU Clark Kent, Who Worked 46.00 Hours this WEEK!
Most hours but no one worked over 40: Peter Parker Worked 39.50 Hours this WEEK!

Sample Run:

```
How many employees are there? 4
Enter Hours Worked, Pay Rate, and Employee name separated by a space: 42 98476.37 Brad Pitt
            Employee Name: Brad Pitt
             Hours Worked:          42.00
                 Pay Rate:       98476.37
               Gross Pay: $4,234,483.91

Enter Hours Worked, Pay Rate, and Employee name separated by a space: 40 12.75 P Diddy
            Employee Name: P Diddy
             Hours Worked:          40.00
                 Pay Rate:          12.75
               Gross Pay:         $510.00

Enter Hours Worked, Pay Rate, and Employee name separated by a space: 108.5 2.47 David Schuessler
            Employee Name: David Schuessler
             Hours Worked:         108.50
                 Pay Rate:           2.47
               Gross Pay:         $427.31

Enter Hours Worked, Pay Rate, and Employee name separated by a space: 37.25 15.5 Tony Stark
            Employee Name: Tony Stark
             Hours Worked:          37.25
                 Pay Rate:          15.50
               Gross Pay:         $577.38


WOW!!!!  What a Dynamo!   David Schuessler Worked 108.50 Hours this WEEK!
```

Details:

Like all programs, you should break this entire process up into several smaller tasks to simplify the design process, make your code more manageable by you and others, and to enhance the reusability of your various methods.

Your main program should call just two methods. The first will prompt the user for how many employees to process and will return the user response (call it getEmployeeCnt). The second will perform the remainder of the operations. Name this second method called from main: processEmployeePay

These two methods are:
getEmployeeCnt:

input to: Scanner as a parameter so the method can read from the console

output: return value of an int which represents the number of employees to process.

processEmployeePay:

input to: Scanner for this method to read employee information while processing.

Integer representing the number of employees to process (can be used for a loop condition).

output: No return value. Does prompt user for input and displays results of processing.

Processing: For Each Employee:

This method will prompt the user to enter each employee's hours, pay rate, and name and will read this data.

It should then call a method that will calculate the gross pay getGrossPay(this method will receive the hours and the payrate, calculate the gross pay using the hours worked and payrate based on the following:

- ✓ all work <u>**over**</u> 48 hours receives double pay rate.
- ✓ work <u>**over**</u> 40 hours <u>**but less than or equal**</u> to 48 hours receives 1.5 pay rate.
- ✓ 40 hours or less receives normal pay rate.

getGrossPay should return the properly calculated gross pay. **Be careful. If there is overtime, only the range of hours of overtime should be calculated at the higher rate.** The remaining time should be calculated at the lower rate(s). E.g. if Bob Doe works 50 hours, 2 hours is at 2 times the rate, 8 hours is at 1.5 times the rate, and the remaining 40 hours is at standard pay.

Call another method named displayEmployeePay which has no return value but is passed the current employee's name, hours, payRate, and grossPay and will output the current employee's information as seen in the Sample Run on page 1.

Check to see if the current employee has the most hours worked and if so, update this information.

Once the processing of all employees is completed a method named displayTopEmployee should be called to display the appropriate information based on how many hours this employee worked.

displayTopEmployee should be passed the topName employee String and the associated topHours. Don't forget to keep track and update these two variables while processing each entered employee.

Method(s) Overview:

Through all the rhetoric above, you might lose sight of all suggested methods to create and call. The following is a list of each:

main

creates a Scanner Object for console input and an integer variable used to hold the number of employees being processed.

continued -------------------- >

main calls getEmployeeCnt (passing it the Scanner object) for a return value representing the number of employees.
main then calls processEmployeePay (passing it the Scanner object and the employee count)

getEmployeeCnt

Receives a Scanner object and prompts the user to input the number of employees and returns this value to the calling program or method.

processEmployeePay

Receives a Scanner object and an integer representing the number of employees.
Processes each employee's gross pay by prompting for the hours, pay rate, and name.
Sends the hours and pay rate of the current employee to getGrossPay in order to receive a gross pay for output.

Calls displayEmployeePay passing it the name, hours, payrate, and grosspay, to be formatted and displayed on the console.

Checks the hours against the topHours and update topHours and topName if needed.

Once all employees are processed processEmployeePay calls displayTopEmployee to display the appropriate message for the employee with the most hours.

getGrossPay

Receives the current employee's hours and pay rate.
Returns the calculated gross pay based on the following conditions:

- ✓ all work over 48 hours receives double pay rate.
- ✓ work over 40 hours but less than or equal to 48 hours receives 1.5 pay rate.
- ✓ 40 hours or less receives normal pay rate.

displayEmployeePay

Receives the current employee's name, hours, pay rate, and gross pay.
Returns nothing
Outputs the current employee's pay information closely formatted to the Sample Run seen on page 1. This is the portions that are indented.
The minor format details you might notice in the gross pay line is not necessary to duplicate exactly. If you can, GREAT but, do not spend a lot of time on this tiny detail.

displayTopEmployee

Receives the top employee name and the top hours. Displays the appropriate message based on the criteria as listed in red print on page one.

The sample run program does implement a space padding method to assist with the output of the gross pay. You are not required to do the same. Just try to come close with your output.

Like all programs, method's purpose should be clearly described in javadoc before each method. Complicated code also needs documentation. Proper use of indentation and braces needs to be consistently implemented.

You should also include multiline non-javadoc documentation at the beginning to identify the course, file name, programming assignment number, the due date, and the instructor. This is followed by a line of space and then just before the class declaration, javadoc to give a brief description of what the program does, an author tag for your name and the current date as a javadoc version tag, such as:

continued --------------------- >

```
/*
 *  Course:               TCSS142 – Introduction to Object-Oriented Programming Summer 2014
 *  File Name:            Assign4.java
 *  Assignment:           4
 *  Due Date:             July 21, 2014
 *  Instructor:           Mr. Schuessler
 */
import java.util.Scanner;
 /**
  * Based on a user input value for the number of employees to process, this program will also prompt the
  * user for each employee’s hours worked, hourly pay rate, and name.  Once enter....  etc.
  *
  * @author your name
  * @version 2014 July 20
  */
```

Don’t forget the import statement to access the Scanner and related classes for keyboard input.
Suggested Steps to Complete this Assignment:

1. Do one thing at a time, compile and make sure it works correctly before moving on.

2. When you work on the processEmployeePay method, don’t worry about the correct gross pay,formatted output, or keeping track of the most pay. You can always add these things later once you’ve tackled the task of inputting the data.

3. Create your getGrossPay method. First, just have it calculate gross pay as if there is no overtime. Once you have that solved (should be easy), call it from processEmployeePay to check that it returns what it should.
   Now add the conditions (Perhaps one at a time or all at once if you feel confident). Test it and if it works, move on to the next method.

4. Write the displayEmployeePay with just general formatting. Call it from processEmployeePay to test it. If all seems well, start adding the formatting features and continue to test until you are satisfied.

5. Now that you have your calculations and formatted output working, go back to the method called processsEmployeePay and try to implement keeping track of the employee’s name and hours who has the most hours (hint: you will need two variable to “remember” this information).

6. You’re almost there, NOW write the displayTopEmployee method. Once you think the logic in your code is correct, call it at the end of processEmployeePay to see how it does. Work out what details remain and hand it in!