Assignment 4 - Employee CS1400 Summer 2011

Description:

Create a new project called **A4_*YourName***. This project includes two files:

1. **Employee.java** // for the Employee class
2. **EmployeeTest.java** // for the main method that tests the Employee class

Employee.java:

Declare a public class called Employee. This is the 'blue-print'. Here we define all the details regarding the Employee class. Let's start by dividing the class body into three regions. We do that by adding the following comments:

// field(s)

// constructor(s)

// method(s)

Now you are ready to add the declarations.

- 3 private fields: name (String), hoursWorked (double), and hourlyRate (double).
- 1 constructor that takes 3 parameters and initializes all three fields
- 6 public methods:
  - getName (get methods have no parameter and return a value whose type matches the field they expose)
  - getHoursWorked
  - setHoursWorked (set methods need a parameter of the same type as the field that gets the new value assigned; they don't return any value)
  - getHourlyRate
  - setHourlyRate
  - grossPay has no parameter and returns a value of type double

How to determine the gross pay:

The company pays straight time for the first 40 hours worked by each employee and 50% more for all hours worked in excess of 40.

```
Examples:
hours worked: 40;  hourly rate: $10 => gross pay = $400
hours worked: 50;  hourly rate: $10 => gross pay = $400 + $150 = $550
```

Notice that the payment depends on a condition. If the number of hours worked is 40 or less there is one way to calculate the payment. Otherwise (if more than 40 hours have been worked) there is another way to calculate the payment.

EmployeeTest.java:

Declare a public class called EmployeeTest.

The sole purpose of this class is to provide the required environment for the main method to exist. Here we are not writing a "blue-print" for a class that will be used as a type later on. There is no need to divide it into the 3 regions. Instead we just declare the main method, where we will write the necessary code to test the Employee class. In order to test the Employee class we do the following:

a) Read in the name, hours worked, and hourly rate.
b) Use the information the user provided to create an object of type Employee (e.g. myEmployee)
c) Print name, hours worked, hourly rate, and gross pay of that newly created employee. (TIP: you'll need the dot operator to do so) Make sure to include the $-sign for money amounts
d) Allow the user to update (change) both the hours worked and the hourly rate.
   You do that by reading in two new values. There is no need to create new local variables to temporarily store the values entered by the user. Instead reuse the local variables you had used for that purpose before.
e) Assign the two new values that the user provided to the corresponding fields of the existing Employee object
f) Repeat c)

Sample Output:

```
name: Alex Johnson
hours worked: 38
hourly rate: 20
Name: Alex Johnson  Hours: $38.0  Rate: $20.0  GrossPay: $760.0

hours worked: 60
hourly rate: 25
Name: Alex Johnson  Hours: $60.0  Rate: $25.0  GrossPay: $1750.0
```

Turning it in

All source code files need a comment with name, course number and assignment on top

Create a zip file, ensure that its name is A4_Y*ourName*.zip, and turn it in via Virtual Campus