ITSE 2321 – OBJECT-ORIENTED PROGRAMMING (JAVA)
Program 10 – Object-Oriented Programming: Inheritance

The owners of the Annan Supermarket would like to have a program that computes the monthly gross pay of their non-exempt hourly employees. The user will enter an employee's first name, last name, the hourly rate of pay, and the number of hours worked for the month, by the week. Assume there are 4 weeks in a month. In addition, Annan Supermarkets would like the program to compute the employee's net pay and overtime pay. **Overtime hours, any hours over 40, are paid at 1.5 the regular hourly rate.** Net pay is gross pay minus taxes (Refer to the tax table on the second page). Use Class Scanner to input the user's data.

Define a class called **HourlyEmployee** that extends the **Employee** Class provided. The class must have **private attributes** to store the total hours worked, the total regular hours worked (sum of hours ≤ 40), and the total overtime hours worked (sum of hours > 40). The class must also have member functions to perform the following tasks:

- A constructor to initialize the first name, last name, and pay rate (See the Employee Class for more details).

- A setter method to set the hours work for the month (**by the week – assume 4 weeks in a month**). Do not write separate setters for total regular and total overtime hours. Use a while/for loop that iterates four times to read the hours worked from the user and call the setter method four times. The header of the method should look like this:

  ```
  public void setTotalHoursWorked(double hoursWorked)
  ```

- A getter method to return
  - the total regular hours work for the month
  - the total overtime hours for the month
  - the monthly regular pay
  - the monthly overtime pay

- A toString method that uses the super class' toString method to display the output which must include the following information:
  - Employee's first and last name
  - Pay rate
  - Total regular hours worked
  - Total overtime hours worked
  - Total hours worked
  - Monthly Regular Pay
  - Monthly overtime pay
  - Monthly gross pay
  - Monthly taxes
  - Monthly net pay

Write a test Class named **Program10** to test the methods in the **Employee** and **HourlyEmployee** Class. Allow the user to run the program as many times as possible until a sentinel name value, **"no"**, has been entered.

No input, processing, or output should happen in the main method. **All work in the test class should be delegated to other non-static methods in the class**.

Every method in your program should be limited to performing a single, well-defined task, and the name of the method should express that task effectively.

Study the Employee Class and understand it before starting the program. Do not modify it. You will not receive credit for Program 10 if you do.

Compile your program and correct all syntax errors and warnings. **You will not receive credit for the program if it does not compile or link successfully.**

All classes in this program must be public, non-static and not nested in other classes.

Tax Table

Bracket	If the gross pay is over	But not over	Tax
1	$0.00	$1,200.00	0%
2	$1,200.00	$2,500.00	10%
3	$2,500.00	$4,500.00	15%
4	$4,500.00	$8,000.00	22%
5	$8,000.00	$10,000.00	28%
6	$10,000.00	$15,000.00	31%
7	$15,000.00	N/A	36%

Run your program **four times** with the data below and save the outputs as one text file named, **Program10-output.txt**.

Run 1

Name: John Doe
Hourly rate: $44.10
Hours worked: 40, 30, 40, 35

Run 2

Name: Jane Doe
Hourly rate: $45.20
Hours worked: 40, 40, 40, 40

Run3

Name: John King
Hourly rate: $15.50
Hours worked: 20, 30, 40, 25

Run 4

Name: <Your Name (First Last) >
Hourly rate: $55.50
Hours worked: 50, 40, 35, 55

Copy and paste the output to a file named **Program10-output.txt**. Create a folder named, **YourFullname_Program10**. Copy your source code and the output file to the folder. Zip the folder and upload it to Blackboard.

Before you upload your program to Blackboard:

- Ensure that your code conforms to the style expectations set out in class and briefly discussed below.

- Make sure your variable names and methods are descriptive and follow standard capitalization conventions.

- Put comments wherever necessary. Comments at the top of each module should include your name, file name, and a description of the module. Comments at the beginning of methods describe what the method does, what the parameters are, and what the return value is. See the **Program1-Template.java** for more details.

- *Program readability and elegance are as important as correctness.* After you have written your method, read and re-read it to eliminate any redundant lines of code, and to make sure variables and methods names are intuitive and relevant.

Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements. **You will not get full credit for this program if it is not written as instructed even if it works as expected.**