**Points: 100**

**Individual Assignment**

**Due Friday, 01/18, at 11:59pm.**

**TASK:**

You want to develop a Java program that will allow you to keep track of a set of employees. In reviewing your employee list, you notice that your employees fall into two categories: Salaried and Hourly. The following table shows the information that you keep in your employee list for each type of employee.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Salaried | Hourly |
| id | int | Yes | Yes |
| name | String | Yes | Yes |
| title | String | Yes | No |
| position | String | No | Yes |
| salary | int | Yes | No |
| hourlyRate | double | No | Yes |

Create a NetBeans project with the prefix Lab101.

Use the same naming convention used in the Lab100 assignment, i.e. Lab101-LastFM.

In this project create three classes named **Employee**, **Salaried** and **Hourly** such that:

* The Employee class contains all of the fields common to both types of entries in your employee list.
* The Salaried class is a subclass of Employee and contains only those fields that are specific to the Salaried entries in your employee book.
* The Hourly class is a subclass of Employee and contains only those fields that are specific to the Hourly entries in your employee book.
* Each of these classes contains all of the “normally expected” methods. The “normally expected” methods are:
  + At least one constructor that is an overload constructor that includes all of the necessary information to create a instance.
  + A getter and setter (accessor and mutator) method for each instance/class variable
  + A toString method
  + An equals method
* Create your classes so that you can keep track of the
  + Total number of Employees
  + Total number of Salaried employees
  + Total number of Hourly employees
  + Note that you are tracking the number of times each class has been instantiated, i.e. a constructor has been called.

Create a fourth class name **Client** that will be used to test your other classes.

In the Client class:

* This class must include the main method
* In the main method
  + Create an array employeeList of type Employee of length 10
  + Add three Salaried contacts to the array
  + Add three Hourly employees contacts to the array
  + Data for each of the contacts must be entered from the keyboard.
  + The employee types must NOT be grouped, i.e. the salaried employees should be interleaved with the hourly employees.
  + Once you entered the data for the six employees:
    - print out the contents of the array using a loop.
    - This loop should print out the contents of every entry in the array including the blank (null) entries.
  + Now give everyone in the employeeList a 10% raise.
    - When applying the raises use a loop to step across the array.
  + After you have given everyone a 10% raise:
    - Print out the contents of the array using a loop.
    - This time do not print the blank (null) entries.
  + Explicitly test the equals methods for each of your classes.

Provide adequate documentation for your code where adequate documentation is defined as follows:

* Each instance or class variable should have a semantically rich name, i.e. the name should tell the reader what the variable represents.
* Each method should include a **Java docs** header.
* Each class should include a Java docs header comment block that includes the following:
  + Your name using the @author tag
  + The date using the @version tag
  + A brief description of the class
* The use of semantically rich identifiers can reduce the amount of documentation that needs to be written.
* Include inline comments to explain what is happening in your code.
* Be sure to comment the Client class as well as the Employee, Salaried and Hourly classes.
* Be sure to remove any unnecessary comments or code, e.g. the comment templates provided by NetBeans.

**HOW TO TURN IN YOUR ASSIGNMENT:**

* You will submit your solution as a single submission on Blackboard.
* Create a complete zip archive of your NetBeans project. This zip archive must be created using the NetBeans File->Export Project->To Zip command.
* **Note that a RAR file is not a ZIP file.**
* Create a Microsoft Word document (must be a .docx file) that
  + Is named following the naming convention used in the Lab100 assignment, i.e. Lab100-LastFM.docx
  + contains the source code for each of your classes including the client class and the output of an example run of your client.
* Submit both your project zip archive and Word document as two separate files in a single Blackboard submission.