Assignment Number 4

100 Points

Prerequisites: Completion of Assignment 3

References: Text chapters 5-9

Skills Required:

1. Object concepts: multiple objects, constructors, constants, private/public, static/instance methods, and methods/constructors overload
2. Arrays of objects
3. Iterations: For, While, Do .. While statements
4. Controls: IF … Else and /or Switch statements
5. Java Enumerator
6. Simple message dialogs

Task Specifications: Create a simple Payroll system that will be used for assignments 5 and 8.

1. The Payroll system scenario basically has four class files, more classes will be added later:
   1. One java enumeration called Status
   2. One class called Employee
   3. One class called PayRecord
   4. One main/entry class called PayRoll
2. **Status Java enumeration:** it is a special data type that has two predefined constants FULLTIME and HOURLY, this will be provided
3. **The Employee class:** 
   1. It has four private data fields/attributes for ID, First Name, Last Name, and EmpStatus of type Status enumeration

**private** **int** eID;

**private** String fName ;

**private** String lName ;

**private** Status empStatus;

* 1. It has one a special constructor with parameters for each of the Employee attributes
  2. it has a public empStstus() method that return the value of EmpStatus field
  3. it has one public toString() method that returns a String of Employee object data. It will return all Employee attributes names and values properly formatted as string

1. **The PayRecord class:**
   1. It has five private field/attributes for ID, Employee object reference, pay hours, pay rate, monthly income, and number of months:

**private** **int** rID;

**private** Employee employee;

**private** **double** payHours;

**private** **double** payRate;

**private** **double** montlyIncome;

**private** **int** numMonths;

* 1. It has three private constants for regular hour payment, overtime hours payments, and tax rate:

**public** **static** **final** **int** *REG\_HOURS* = 40;

**public** **static** **final** **double** *OT\_RATE* = 1.25;

**public** **static** **final** **double** *TAX\_RATE* = 0.05;

* 1. It has two special constructors with the same name and different parameters for each of the Employee attributes based on the user selections of the Employee status of either FULLTIME or HOURLY (this is known as method overload):
     1. This constructor will be used for Employee with HOURLY status:

PayRecord(**int** id, Employee e, PayPeriod period, **double** hours, **double** rate){}

* + 1. This constructor will be used for Employee with FULLTIME status:

PayRecord(**int** id, Employee e, PayPeriod period, **double** mIncome, **int** mNum){

* 1. It has one public payGross() method that calculates and returns the gross pay of the Employee object based on its status of FULLTIME or HOURLY using the empStstus() method in Employee class.
     1. For the full time employee: the gross pay is simply the number of months multiplied by the pay income per months; for example, 3 months with $1500 per month: the gross pay = $4500.
     2. For the hourly employee: the gross pay is simply the number of hours (up to and including 40 hours) multiplied by the pay rate; for example, 40 hours with $10/hour: the gross pay = $400. For the amount over 40 hours is paid at “time and a quarter” (1.25, *OT\_RATE*); for example, 50 hours with $10/hour: the gross pay = $525, $400 for the first 40 hours and $125 with extra hours
  2. It has one public netPay() method that calls grossPay() method to calculates and returns the net payment after tax which is 5% (*TAX\_RATE*) that is even for both employee types
     1. The net pay is simply calculated by subtracting the gross pay from the tax that is obtained by multiplying the gross pay by the tax rate. Use the grossPay() method that you created above rather than re-calculating it here. For example, if the gross pay is $1000 and the tax rate is 5%, the net pay is $950.
  3. It has one public toString() method that returns PayRecord object data as string. . It will return all PayRecord attributes names and values formatted as string including the net pay value using netPay() method. For Employee data, call its toString() method.

1. **The main/entry PayRoll class :**
   1. It has the two public static field/attributes for arrays of Employee object references and PayRecord object references:

**public** **static** Employee[] *employees*;

**public** **static** PayRecord[] *payRecords*;

* 1. It has one public static constant for the number of Employe and PayRecord references:

**public** **static** **final** **int** *NUM\_PAY\_EMP\_RECORDS* = 3;

* 1. It has one public static createEmployee() method that creates a new Employee object and adds it to *employees* array
  2. It has one public static createPayRecord() method that creates a new PayRecord object based on the Employee status FULLTIME or HOURLY and adds it to *payRecords*  array.
  3. It has the **public** **static** **void** main(String[] args) method that interact with user to input employee attributes and their pay records using message dialogs. The program loops/iterates three times (using constant *NUM\_PAY\_EMP\_RECORDS*) for employee and pay record inputs:
     1. Use (one message dialog) for each employee input and then calls createEmployee() with user input data parameters to create the employee object, then shows a message box that confirms employee object is created . The input employee data includes on mix of FULLTIME employee and/or HOURLY employee based on the user selections using a proper message dialog and control statements. Use the empStatus attribute in Employee constructor to determine the employee status using Java enum Status class such as: *employees*[i] = **new** Employee(id,firstName, lastName, Status.*HOURLY*); or *employees*[i] = **new** Employee(id, firstName, lastName, Status.*FULLTIME*);
     2. Use (one message dialog) for each pay record input (PayRecord) of the current entered employee and then calls createPayRecord() with user input data parameters to create the PayRecord object, then shows a message box that confirms PayRecord object is created and displays the PayRecord data using toString() method of PayRecord class.
     3. The program terminates with a proper message box prompting the user that the program is ended.
     4. All message boxes and message dialogs should ONLY be inside the **public** **static** **void** main(String[] args) method

1. Evaluation Criteria
2. All tasks must be completed to receive credit for this assignment
3. Program should report the correct values