Assignment Number 5

100 Points

Prerequisites: Completion of Assignment 4

References: Text chapters 5-9

Skills Required:

1. Advanced Java Graphical User Interface techniques
2. Buttons, Labels, and other controls
3. JtextAreas, and other fields
4. Listeners and events

Task Specifications: Complete the Payroll system that has been created in Assignments 4.

1. The Payroll system scenario will have three more classes:
   1. One class called PayPeriod,
   2. One class called WithHoldings
   3. One main/entry class called RunPayRoll
2. **The PayPeriod class:** 
   1. It has three private data fields/attributes for ID, pay start date, and pay end date :

**private** **int** pID;

**private** Date pStartDate, pEndDate;

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

1. **The WithHoldings class:**
   1. It has three private constants for hourly tax rate, state tax rate, and federal tax rate:

**public** **static** **final** **double** *HOURLY\_TAX* = 0.05;

**public** **static** **final** **double** *STATE\_TAX* = 0.12;

**public** **static** **final** **double** *FEDRAL\_TAX* = 0.7;

* 1. It has one public static getHourlyTax(double grpssPay) method that calculates and returns the tax of the hourly Employee gross pay:
     1. It has one parameter for the employee gross pay and uses this parameter to calculate the tax by simply multiply the grossPay value by the constant hourly tax rate (*HOURLY\_TAX*) and return the result as double type
  2. It has one public static getFullTimeTax(double grpssPay) method that calculates and returns the tax of the full time Employee gross pay:
     1. It has one parameter for the employee gross pay and uses this parameter to calculate the tax by simply multiply the grossPay value by the constant hourly tax rate (*STATE\_TAX*) added with multiplying the grossPay value by the constant hourly tax rate (*FEDRAL\_TAX*) and return the result as double type.

1. **Update PayRecord Class:**
   1. Add one private data fields/attributes of object reference for PayPeriod to the PayRecord attributes:

**private** PayPeriod payPeriod;

* 1. Delete the private constants for tax rate from the PayRecord attributes.
  2. Update netPay() method to use Class WithHoldings class to calculate the tax for FULLTIME employee and HOURLY employee:
     1. You need to use *if .. else* statement or *switch* statement to check whether employee is FULLTIME or HOURLY employee
     2. For FULLTIME employee you need to call WithHoldings. getFullTimeTax() and pass the paygross() to calculate the tax and then subtract the tax from the grossPay to get the net pay.
     3. For HOURLY employee you need to call WithHoldings. getHourlyTax() and pass the paygross() to calculate the tax and then subtract the tax from the grossPay to get the net pay.
  3. Update public toString() method to include PayPeroid data object by using the public toString() method of PayPeroid.

1. **The PayRoll class :**
   1. Use the template provided called GUIDemo in order to update PayRoll class to create a graphical application for PayRoll system, for example, as shown below. It is up to the student to create a preferred graphical application design, use your creativity
   2. Add the required import packages to create a user GUI as in the GUIDemo class.
   3. There will be a number of fields, labels, buttons, and a text area based on the student’s preference names.
   4. Delete the main method
   5. Change the class name/signature as follows:

**public** **class** PayRoll **extends** JPanel **implements** ActionListener

* 1. Add a default constructor with no parameters that will call the required methods when the PayRoll object is instantiated and it adds the action listeners to buttons, see the following example of the constructor that still needs more action listeners for buttons:

**public** PayRoll () {

makeTheObjects();

doTheLayout();

addEmployeeButton.addActionListener( **new** java.awt.event.ActionListener(){

**public** **void** actionPerformed(ActionEvent e){

createEmployee\_actionPerformed(e);

} });

CloseButton.addActionListener( **new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(ActionEvent e){

close\_actionPerformed(e); } });

} // end of constructor

* 1. Add empty body method: **public** **void** actionPerformed(ActionEvent e){}
  2. Add the close\_actionPerformed(), makeTheObjects(),doTheLayout() methods from GUIDemo and update them to match your design and input data
  3. Change the signature of the createEmployee() method from ***public******static******void*** *createEmployee(){}* to ***void*** *createEmployee\_actionPerformed(ActionEvent e){}* and keep the code from Assignment 4 in this method as it is and add the validation code as needed. Follow the same procedure for all the other buttons of adding pay record object.
  4. The public createPayRecord()method should now create object of PayPeriod and added it to the PayRecord Constructor parameters similar to the other parameters.
  5. **Validate** the input date format and the pay roll period should be at least one month, an employee can be paid for at least one month.
  6. As in Assignment 4, after adding 3 records “*NUM\_PAY\_EMP\_RECORDS*”, the program should show in the text area added pay record data including pay period
  7. The program should not crash for invalid data inputs neither should accept empty inputs

1. **The RunPayRoll class:**
   1. Use the template provided called RunGUIDemo
   2. The class name should be **class** RunPayRoll **extends** JFrame
   3. The main method should create objects of proper types by changing these statements:

JFrame f = **new** RunPayRoll();

contentPane.add( **new** PayRoll());

1. Evaluation Criteria
2. All tasks must be completed to receive credit for this assignment
3. Application should perform correct calculations
4. There should be no screen artifacts left from previous uses
5. The application should not crash from improper input
6. The application should notify the user of improper inputs or empty text fields
7. Application should stop based on user closing the window, and
8. Application should stop based on user pressing “Close.”

