Project Assignment 3

75 Points

References: Referenced Textbook and Week 2, 3, 4, 5, 6, 7 & 8 handouts

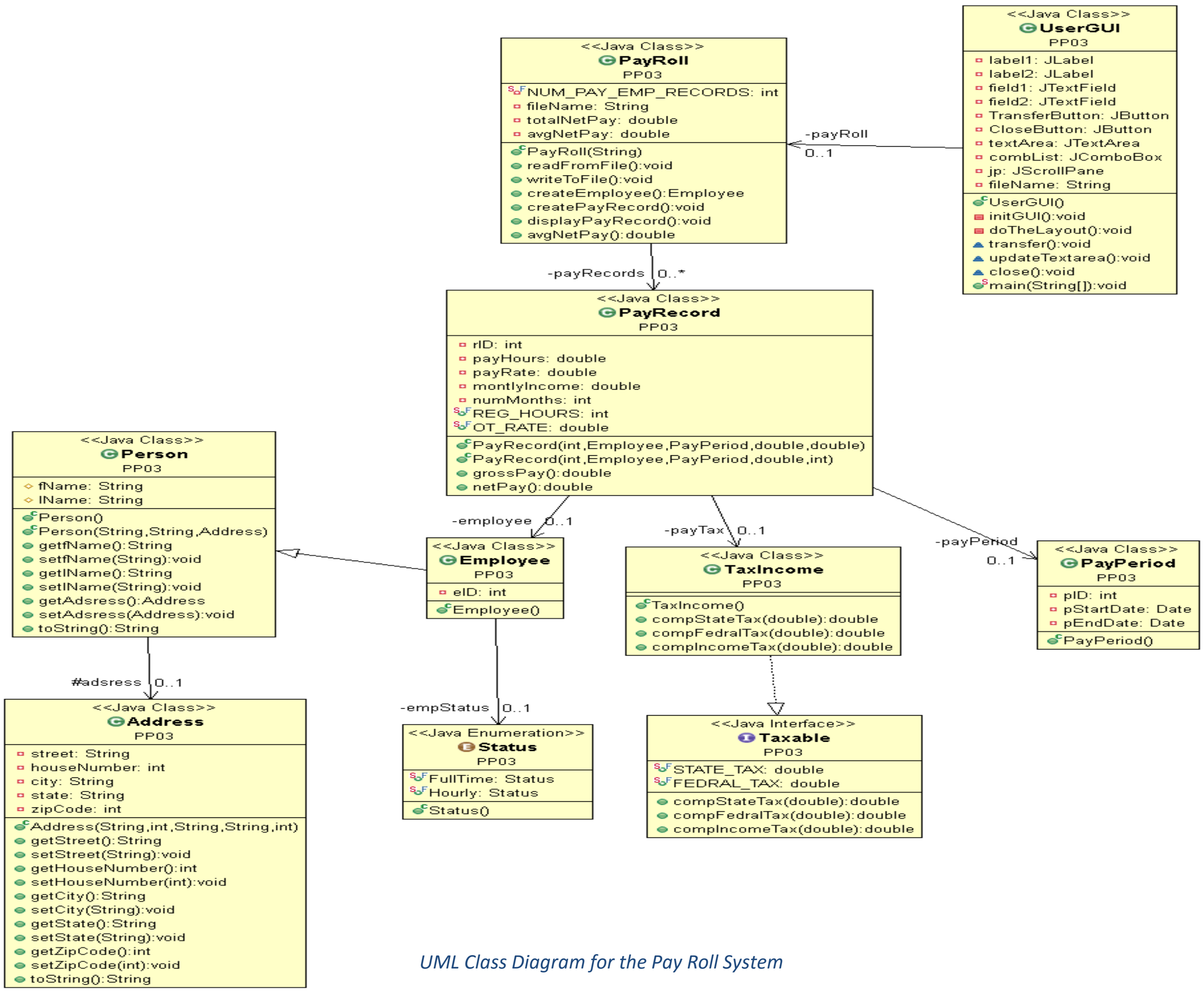
Due date: Monday, Mar 27 2017, by 11:59pm

Skills Required:

1. Advanced Java Graphical User Interface techniques and user input interaction
2. GUI controls; JLabel, JTextField, JButton, JTextArea, JComboBox, JScrollPane, etc
3. The use of Java action listener and event handler classes
4. Input validations and the use of checked/unchecked exception classes
5. JOPtionPane dialog message boxes
6. Inheritance and polymorphism
7. Abstract classes and interfaces
8. Files

Description:

1. Create a new java project and name it PP3.



UML Class Diagram for the Pay Roll System

2. Download the files from the associated source folder and add these files to the PP3 project
3. This project implements a Payroll system to calculate the monthly/hourly pay for at most 10 employees. The program must prevent users from adding more than 10 employees.
4. The program must terminate after calculating the pay for 10 employees or when the user press the “Close” button
5. The payroll system has initial payroll data stored in a text file (PayRoll.txt) that is used to instantiate some program objects
6. The system stores or appends complete payroll record information including the calculated gross pay, tax, and a net pay in the PayRecord.txt file
7. The payroll system GUI interface presents information to a user based on the data entered by the user
8. The system allows a user to add employee, pay period, and pay record information, and shows a full employee pay record in the GUI, and stores the new employee pay record into the PayRecord.txt file

Task Specifications:

1. The UML class diagram model shown above represents the design of the project
2. The project has 8 business logic classes, one interface, and one class GUI
3. Code must be written to implement the project classes and logic as specified in the given UML class diagram model, as well as in the *associations* between classes
4. None of the attributes of the classes can accessed from any external classes
5. The methods in the UML class diagram must be implemented as it will be described below
6. Once the system starts, the readFile() method in the PayRoll.java class is invoked to read employees data from the PayRoll.txt file. Hence, the program instantiates some project objects, computes employees’ pay records, and updates the GUI text area. The PayRoll.txt file has data for three employees.
7. The user interaction with GUI results in method calls to the PayRoll.java class, which in turn call other methods from other class objects to implement the payroll program requirements, such as making calculations, and/or viewing an employee pay record
8. Update the code of the provided UserGUI.java class to create a graphical interface for the payroll system, as shown in the snapshots below
9. No business logic code should be added to the UserGUI.java class, all the business logic must be implemented in the business logic classes such as the PayRoll.java class. The graphical interface must have all the controls as shown, this includes some GUI components: JTextField, JLabel, JTextArea, JComboBox, JRadioButton, etc.
10. Use your own creativity to create the system GUI by adding some controls and/or enable or disable some controls as needed at the run time
11. You must implement the action listeners for all add buttons in project GUI
12. The action listeners for the "Add Employee" and "Add Pay Record" buttons must implement the following:
    1. When clicking on the "Add Employee" and "Add Pay Record" buttons, the program must read and validate the data text fields before adding new objects
    2. When clicking on the "Add Employee" and "Add Pay Record" buttons, the program will use the business logic in the PayRoll.java class to create system objects and updates pay records array and the GUI text area

13. Validate the number of months in the months' text box so that the value does not exceed the months in the start and end pay period dates.
14. Validate the input date format, and validate that the payroll period is at least one day. An employee can be paid for at least one working day.
15. The program must not crash for any invalid data inputs neither should accept empty inputs
16. After successfully adding a new pay record, the program will show the added pay record, pay period, and employee object data in the text area
17. The program must also display the current total and average net pays for all added employees' pay records
18. The Payroll system scenario has 8 business logic classes and one interface:
    1. A java enumeration called Status, *provided*
    2. Class Employee, a subclass of Class Person
    3. Class Person, *provided*
    4. Class Address, *provided*
    5. Class PayRecord
    6. Class PayRoll
    7. Class PayPeriod
    8. Class TaxIncome, implements interface Taxable
    9. Interface Taxable, *provided*
19. **Status Java enumeration:** it is a special data type that has two predefined constants FULLTIME and HOURLY, this will be provided
20. **The Employee class:**
    1. It extends superclass Person, superclass Person uses class Address
    2. In addition to the inherited data fields from superclass Person, it has two data fields/attributes for ID and EmpStatus of type Status enumeration:
```
private int eID;
private Status empStatus;
```
    3. It has one a special constructor with parameters for each of the Employee attributes
    4. It has a public empStatus() method that return the value of empStatus field
    5. It has public getter/setter methods for the Employee class data fields
    6. 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
21. **The PayRecord class:**
    1. It has five private field/attributes for ID, Employee object reference, pay period object, pay hours, pay rate, and monthly income:
```
private int rID;
private Employee employee;
private PayPeriod payPeriod;
private Tax payTax;

private double payHours;
private double payRate;

private double montlyIncome;
private int numMonths;
```
    2. It has two private constants for the regular hour payment and overtime hours payment:
```
private static final int REG_HOURS = 40;
private static final double OT_RATE = 1.25;
```

3. 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):
   3.1 This constructor will be used for Employee with HOURLY status:
   ```
   PayRecord(int id, Employee e, double hours, double rate){}
   ```
   3.2 This constructor will be used for Employee with FULLTIME status:
   ```
   PayRecord(int id, Employee e,  double mIncome, int mNum){}
   ```
   3.3 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 empStatus() method in Employee class.
      3.3.1 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.
      3.3.2 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
   3.4 It has one public netPay() method that calls grossPay() method to calculates and returns the net payment for both employee types
      3.4.1 The net pay is simply calculated by subtracting the gross pay from the pay tax that is obtained by calling the Tax.java class method to calculate the pay tax. Use the grossPay() method that you have created rather than re-calculating the gross pay here. For example, if the gross pay is $1000 and the tax pay is $100, then the net pay is $900.
   3.5 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 calculated gross pay, income tax, and net pay, it also concatenate the Employee and PayPeriod objects data. For Employee and PayPeriod objects data, call their toString() method here.

22. The PayRoll class:

1. It has the following public field/attributes:
   ```
   private String fileName;
   private PayRecord[] payRecords;
   private  double totalNetPay;
   private  double avgNetPay;
   ```
   3.1 It has one private static constant for the number of Employe and PayRecord references:
   ```
   public static final int NUM_PAY_EMP_RECORDS = 10;
   ```
   3.2 This class has one custom constructor that needs to be called to create a PayRoll object
   ```
   public PayRoll(String fileName){}
   ```
   3.3 It has a public void readFromFile() method that reads data from PayRoll.txt file and calculates the employee pay for each employee in this file, and stores the employee pay records in the PayRecord.txt file using the public writeToFile() method
   3.4 It has a public void writeToFile() method that writes a PayRecord object data to the PayRecord.txt file

3.5 It has one public Employee createEmployee() method that creates a new Employee object and returns the employee object

3.6 It has one public void createPayRecord() method that creates a new PayRecord object based on the Employee status FULLTIME or HOURLY, and it adds the PayRecord object to *payRecords* array

3.7 It has a public displayPayRecord() method that will be called from the GUI class to display the current employee pay record on the GUI text area

23. The PayPeriod class:

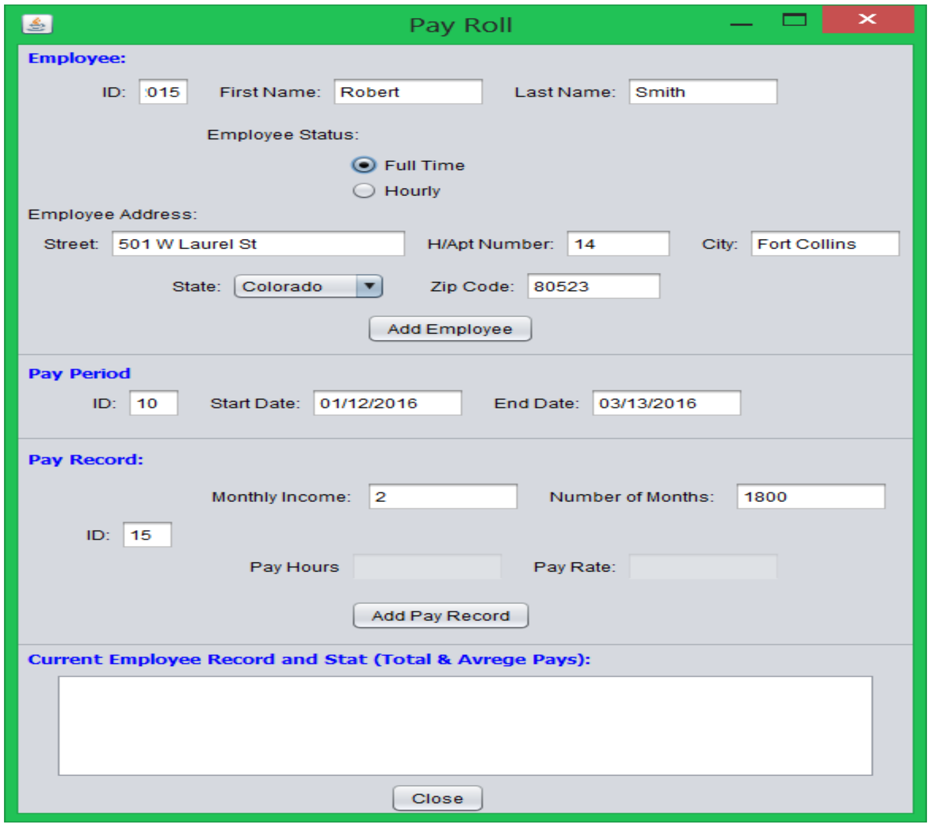
1. It has three private data fields/attributes for ID, pay start date, and pay end date:

```
private int pID;
private Date pStartDate, pEndDate;
```

2. It has one a special constructor with parameters for each of the PayPeriod attributes
3. 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
4. Add setter/getter methods for the data fields

24. The TaxIncome class:

1. It implements interface Taxable
2. It has a default constructor
3. It implements all unimplemented methods in the Taxable interface in order to compute the income tax from the state and federal taxes.



Employee: ID: 015 First Name: Robert Last Name: Smith
Employee Status: [✓ Full Time] Hourly
Employee Address:
Street: 501 W Laurel St H/Apt Number: 14 City: Fort Collins
State: Colorado Zip Code: 80523
Pay Period ID: 10 Start Date: 01/12/2016 End Date: 03/13/2016
Pay Record: Monthly Income: 2 Number of Months: 1800
ID: 15
Pay Hours Pay Rate:

Evaluation Criteria:

1. All tasks must be completed as specified to receive a full credit
2. Application must perform all the requirements correctly, including read from and write to data file

3. The application must not crash from improper inputs or any user's interactions
4. For validation, the application must notify the user of improper inputs or empty text fields
5. All classes and their relationships must be implemented as specified in the UML class diagram model
6. There must not be any business logic code in the GUI class
7. Application must stop on pressing button "Close"

Submission:

Copy the .java source files from the *src* folder in your *work space* to another folder that must be named following the provided naming format in this course, then zip and upload the file under this assignment answer in Canvas. You must read the project demo document in Canvas before the review time.

File Name: *FLLLLPP3.zip (F = first letter in your first name and LLLL = your last name)*

Grading Rubric – PP03

Name: ____

Requirements	**Comments**	**Max Points Allocated**	**Points Earned**
General Code Structure: Coding of all the expected Java classes (e.g. 8 Business logic and GUI class, and the association between the classes) with the expected methods (5) Use of Java comments in the source code (2) Indentations, good variable names/class members (2) Successful compilation (no compile error) and running of code (able to execute the program) (6)		15	

Exception handling of the input values (e.g., program will not crash is no value is entered, empty space is entered, invalid value is entered) (4) The program must read and validate the data entered in the GUI, before creating new objects and storing in the text file (6) Validate the number of months in the months' text box, input date format, payroll period is at least one day (4) Use of Buttons, Labels, JtextAreas, radio buttons, and other controls and display the full employee pay record in the GUI (6) Formatting of the output values e.g, date with slash and proper legible display of the values in the Current employee and Stat area (3) Close button (2) File I/O such as reading from the text file and adding the employee and the pay record in the text file (10)		30	
Functionality: The program must prevent users from adding more than 10 employees (2) Adding employee including the use of listener and event class (3) Adding pay record including the use of listener and event class (3) Logic to display of the current employee record and the stat (2) Computation of various payroll record information e.g., gross pay, net pay (5) Computation for the full time and hourly logic (3) Use of enumeration (2) Proper use of coding standards, e.g, object hierarchy including extend, interface implementation, use of constructor, private qualifier (5)		30	
Total		**75**	

Total: _____/75