Project 5: Payroll (Part 1)

CS 1410

Background

In this project you will implement a simple payroll system. For the first part of the assignment, you will submit only a UML class diagram.

The hypothetical company we are considering has 3 **classifications** of employees:

1. Hourly
2. Salaried
3. Commissioned

There are 24 pay periods per year; $1/24^{th}$ of a salary is paid each semi-monthly pay period to employees who receive a salary. We won't worry about taxes and other deductions for this assignment.

Commissioned employees are **Salaried** employees, but also receive an additional payment of their total sales times their commission rate.

Hourly employees are simply paid their hourly rate times their hours worked for that pay period. We won't worry about overtime.

Employees can have their classifications **changed** during their employment!

We will simulate issuing payment by just writing text to a file. The important part of this project is the object-oriented **design**. You will do this assignment in two parts:

1. Submit a UML class design document for your classes.
2. For Part 2, submit your source file, *payroll.py*, as well as *payroll.txt* and *payroll_old.txt*, as explained in Part 2 of this assignment in a separate document.

Your UML class diagram should have all of the attributes and methods that you will use in your implementation for Part 2 of this assignment.

Requirements (Part 1)

Employee objects have the following required attributes:

- **emp_id**: string
- **first_name**: string
- **last_name**: string
- **address**: string
- **city**: string
- **state**: string
- **zipcode**: string
- **classification**: a concrete instance of either **Hourly**, **Salaried**, or **Commissioned**

The applicable CRC cards appear below:

Employee	
• Manage Employee attributes • Change employee's classification • Initiate payment to employee	• Classification • PaymentMethod

Abstract **Classification**	Hourly, Salaried
• An abstract base class for the three types of classifications. • Specifies the abstract method issue_payment	

Hourly	Classification
• Know the employee's hourly_rate • Add/store new time cards • Compute the Hourly employee's pay	

Salaried	Classification Commissioned
• Know the employee's salary • Compute the Salaried employee's pay	

Commissioned	Salaried
• Know the employee's commission rate • Add/store new receipts • Compute Commissioned employee's pay (includes salary)	

The following data files are provided:

- *employees.csv*
- *timecards.csv*
- *receipts.csv*

The **csv** file is a text file containing employee attributes, separated by commas. Here are the first few lines:

```
id,first_name,last_name,address,city,state,zip,classification,salary,commission,hourly
51-4678119,Issie,Scholard,11 Texas Court,Columbia,Missouri,65218,3,134386.51,34,91.06
68-9609244,Jed,Netti,85 Coolidge Terrace,San Antonio,Texas,78255,2,159648.55,47,45.7
47-2771794,Galvan,Solesbury,3 Drewry Junction,Springfield,Illinois,62794,2,91934.89,39,47.92
11-0469486,Reynard,Lorenzin,3233 Spaight Point,Houston,Texas,77030,1,87578.56,27,86.84
41-5906217,Sandi,Bernardinelli,24129 Susan Junction,Houston,Texas,77080,1,75273.14,44,58.58
62-5530062,Shelli,Gosnold,0348 Dakota Hill,Pasadena,California,91199,2,139728.46,30,65.55
87-6664864,Deonne,Plane,61894 Westerfield Lane,Portland,Oregon,97296,3,103514.53,36,92.61
07-3318962,Filia,Cramb,2 Hallows Parkway,Fort Worth,Texas,76192,3,138257.72,20,99.59
80-1251478,Berni,Whittam,77015 Prairie Rose Road,Chicago,Illinois,60669,2,46020.03,40,43.25
38-2406306,Kelby,Shaxby,5 Roxbury Place,Dallas,Texas,75231,2,46173.64,31,75.03
06-4413296,Carola,Doche,820 Blackbird Crossing,Houston,Texas,77234,3,136114.4,29,65.06
…
```

This file must be read carefully. The first employee, Issie Scholard with ID ’51-4678119, is an hourly employee (classification = 3) making $91.06 per hour! Jed Netti is commissioned (classification = 2) at a rate of 47% on top of a salary of $159,648.55. Raynard Lorenzin (classification = 1) receives a salary of $87,578.56.

Not all of the last 3 attributes are used for each employee. For example, for Issie, only the hourly rate applies, and is stored in his/her Hourly classification attribute. For Raynard, only the Salary field amount is stored in his Salaried **classification** object. All **classification** objects contain a reference to the Employee object so the employee’s information is available when payroll is run. Do not use a CSV module to process this file; just use **split** on each line as you read it from *employees.csv*.

Here is the main program you should use (also appears in the file *p5.py* in Canvas; use it as-is):

```
''' p5.py: Illustrates the payroll module. '''

from payroll import *
import os, os.path, shutil

def main():
    load_employees()       # You will implement the first three of these functions
    process_timecards()
    process_receipts()
    run_payroll()          # This function is given to you in Part 2

    # Save copy of payroll file; delete old file
    shutil.copyfile(PAY_LOGFILE, 'paylog_old.txt')
    if os.path.exists(PAY_LOGFILE):
        os.remove(PAY_LOGFILE)    # You define PAY_LOGFILE = ‘paylog.txt’ globally

    # Change Issie Scholard to Salaried by changing the Employee object:
    emp = find_employee_by_id('51-4678119')
    emp.make_salaried(134386.51)
    emp.issue_payment()

    # Change Reynard,Lorenzin to Commissioned; add some receipts
    emp = find_employee_by_id('11-0469486')
    emp.make_commissioned(50005.50, 27)
    clas = emp.classification
    clas.add_receipt(1109.73)
    clas.add_receipt(746.10)
    emp.issue_payment()

    # Change Jed Netti to Hourly; add some hour entries
    emp = find_employee_by_id('68-9609244')
    emp.make_hourly(47)
    clas = emp.classification
```

```
    clas.add_timecard(8.0)
    clas.add_timecard(8.0)
    clas.add_timecard(8.0)
    clas.add_timecard(8.0)
    clas.add_timecard(8.0)
    emp.issue_payment()

if __name__ == '__main__':
    main()
```

The functions called from **main** are defined outside of any class at the module level.

The **load_employees** function reads the contents of *employees.csv* and creates a list of employees at the module level, populating each Employee instance with the correct type of Classification. The list of employees needs to be available in multiple functions, hence it must be at the module level.

The **process_timecards** function reads *timecards.csv* and adds each hourly record to a list of floats representing the hours worked in the Hourly employee's Hourly classification object. The *timecards.txt* file contains the IDs of hourly employees and their timecard entries:

```
51-4678119,7.6,3.1,1.4,4.1,6.4,7.7,6.6
87-6664864,5.2,3.8,7.5,7.8,7.0,2.1,3.6,7.1,5.1
07-3318962,2.5,6.9,7.9,7.7,1.1,2.6,5.9,6.1,7.5,3.6
06-4413296,5.9,4.2,6.2,3.1,2.2,7.6,4.7,6.7,1.9,7.4
…
```

Not all hourly employees will have current timecard data, and the number of entries may vary. Do not make a payment if there are no entries.

The **process_receipts** function behaves analogously for Commissioned employees using the file *receipts.csv*:

```
68-9609244,109.66,835.97,631.92,862.67,732.21
47-2771794,912.48,873.15,884.0,429.34
62-5530062,465.92,291.14,941.82,213.59
38-2406306,770.1,604.38,742.86
…
```

Not all commissioned employees will have current receipt data, and the number of receipts may vary. If there are no receipts for a commissioned employee, they just receive their salary, otherwise, they receive their salary plus commission, which is their commission percentage times the sum of the amounts on their receipts.

Implementation Notes

This is the most complex and most important project of the semester. Plan your time wisely. You will spend most of your in the implementation, so finish your UML class diagram early. The CRC cards above will help you, as will the functions and methods illustrated in **main** above, most of which you will implement.

FAQs

Q. Why did you give part of the answer away by explaining the relationship between the abstract Classification class and its subclasses? I mean, now we know how to draw the UML diagram.
A. I'm a nice guy. Plus, this is probably your first time drawing your own UML class diagram, so I wanted to give you a head start. It is still a good exercise.

Q. What is the purpose of the Classification class?
A. To show that the concrete classification classes, Hourly, Salaried, and Commissioned, are related, and also to explicitly define an abstract **compute_pay** method for the concrete classes to override.

Q. What is the relationship between Classification and the concrete classification classes.
A. Each of the concrete classes "**is-a**" Classification, which indicates inheritance. Note that Commissioned "is-a" Salaried.

Q. What is the relationship between Employee and Classification?
A. An Employee object "**has-a**" concrete Classification object, either Hourly, Salaried, or Commissioned.

Q. How is the pay actually calculated?
A. When a user calls upon an Employee object to issue payment, the **Employee.issue_payment** method will call upon the Employee's classification object to actually compute and return the payment amount. Then **issue_payment** prints an entry to the *payroll.txt* file.

Q. What about those global functions like **load_employees**, **process_timecards**, etc.?
A. Include those in a Note widget in your UML diagram so everything is in one place.