Programming Project D

Payroll

Background

In this project you will implement a simple payroll system. The hypothetical company we are considering has 3 classifications of employees:

1. Hourly
2. Salaried
3. Commissioned

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

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

Salaried employees are simply paid 1/24th of their salary.

Commissioned employees are Salaried employees, but also receive an additional payment of their total sales times their commission rate (a percentage).

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

Payment is distributed in two ways:

1. Direct transfer to a bank account
2. Receiving a check by mail

Employees can **change** their desired payment method at any time.

We will simulate issuing payment by just writing text to a file. The important part of this project is the object-oriented **design** and **implementation**.

Requirements

Employee objects have the following required attributes:

- **emp_id**: string
- **name**: string
- **address**: string
- **city**: string
- **state**: string
- **zipcode**: string
- **classification**: a concrete instance of either Hourly, Salaried, or Commissioned
- **paymethod**: a concrete instance of either DirectMethod or MailMethod

The CRC cards below provide describe the classes you are to implement.

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

Abstract **Classification**	Hourly, Salaried
• Specifies the abstract method issue_payment	

Hourly	Classification
• Know the employee's hourly_rate • Add new time cards • Hold current time cards • Compute the employee's pay	

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

Commissioned	Salaried
• Know the employee's salary, and commission rate • Add receipts • Hold the current sales receipts • Compute employee's pay	

Abstract **PaymentMethod**	DirectMethod, MailMethod
• Specifies the abstract method issue, which posts the employee's payment • Knows the employee	• Employee

DirectMethod	PaymentMethod
• Know the employee's bank routing and account numbers • Transfers funds to the bank (in our case, just prints a line to the log file)	• Employee

MailMethod	PaymentMethod
• Know the employee's name and complete address • Print paycheck (in our case, just prints a line to the log file)	• Employee

You will do this assignment in two parts:

1. First, submit a UML class design document for your classes. (**draw.io** is a nice free tool)
2. Then submit a source file, *payroll.py*, with your implementation.

The UML class diagram should contain all attributes and methods for each class.

The following data files are provided:

- employees.csv
- timecards.txt
- receipts.txt

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

```
ID,Name,Address,City,State,Zip,Classification,PayMethod,Salary,Hourly,Commission,Route,Account
688997,Karina Gay,998 Vitae St.,Atlanta,GA,45169,1,1,45884.99,46.92,34,30417353-K,465794-3611
522759,TaShya D. Snow,2624 Hendrerit St.,College,AK,99789,2,2,50005.50,68.13,25,36644938-8,244269-0000
983010,Jolene Burgess,P.O. Box 873,South Burlington,VT,32036,2,2,20042.77,40.17,23,15300058-1,828625-2906
939825,Yoko M. Pitts,4825 Nec Ave,Meridian,ID,45614,1,1,35251.89,46.64,13,44553589-3,785957-2104
379767,Jin W. Morrison,8628 Id St.,Milwaukee,WI,80356,3,1,64467.10,82.98,33,21038669-6,654904-8491
…
```

This file must be read carefully. The first employee, Karina Gay with ID ‘688997’, is an hourly employee (classification = 1) and receives her pay by direct deposit (paymethod = 1). TaSha D. Snow is salaried (classification = 2) and has her check mailed to her address (paymethod = 2). Jin Morrison is commissioned (classification = 3) and on direct deposit.

Not all of the last 5 attributes are used for each employee. For example, for Karina, only the hourly rate, route (bank routing number) and account attributes are used; the hourly rate is stored in her Hourly classification attribute and the bank routing and account numbers are stored in her DirectMethod **paymethod** attibute. For TaSha, only the Salary field is stored in her Salaried **classification** object. All **paymethod** objects contain a reference to the Employee object so the employee's information is available when payroll is run. You can either use a CSV module to process this file or just use **split** for each line.

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

```python
from payroll import *
import shutil

def main():
    load_employees()
    process_timecards()
    process_receipts()
    run_payroll()

    # 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)

    # Change Karina Gay to Salaried and MailMethod by changing her Employee object:
    emp = find_employee_by_id('688997')
    emp.make_salaried(45884.99)
    emp.mail_method()
    emp.issue_payment()

    # Change TaShya Snow to Commissioned and DirectMethod; add some receipts
    emp = find_employee_by_id('522759')
    emp.make_commissioned(50005.50, 25)
    emp.direct_method('30417353-K', '465794-3611')
    clas = emp.classification
    clas.add_receipt(1109.73)
    clas.add_receipt(746.10)
    emp.issue_payment()

    # Change Rooney Alvarado to Hourly; add some hour entries
    emp = find_employee_by_id('165966')
    emp.make_hourly(21.53)
    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 and PayMethod. 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.txt* 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:

```
688997,5.0,6.8,8.0,7.7,6.6,5.2,7.1,4.0,7.5,7.6
939825,7.9,6.6,6.8,7.4,6.4,5.1,6.7,7.3,6.8,4.1
900100,5.1,6.8,5.0,6.6,7.7,5.1,7.5
969829,6.4,6.6,4.4,5.0,7.1,7.1,4.1,6.5
283809,7.2,5.8,7.6,5.3,6.4,4.6,6.4,5.0,7.5
224568,5.2,6.9,4.2,6.4,5.3,6.8,4.4
163695,4.8,7.2,7.2,4.7,5.1,7.3,7.5,4.5,4.6,7.0
454912,5.5,5.3,4.5,4.3,5.5
285767,7.5,6.5,6.3,4.7,6.8,7.1,6.6,6.6
674261,7.2,6.2,4.9,6.5,7.2,7.5,5.0,7.9
426824,7.4,6.5,5.7,8.0,6.9,7.5,6.5,7.5
934003,5.8,7.5,5.8,4.8,5.9,4.8,4.0,6.6,5.5,7.2
…
```

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

```
165966,241.34,146.55,237.48,96.37
379767,128.80,121.98,66.99,168.72
265154,240.20,83.69,52.31,77.29,142.12
160769,63.02,163.42,140.06,84.15
…
```

Implement the five module-level functions used in the **main** function. Use *payroll.py* as your main module.

After running your program, compare the three new entries in paylog.txt to paylog_old.txt that was copied earlier in main, and verify that the pay amount and method have been changed appropriately for the three employees updated in main. Submit your payroll.py, paylog_old.txt, and paylog.txt output files for part 2 of this assignment.

Implementation Notes

Every time **run_payroll** executes, first delete the previous *paylog.txt* file, if it exists (the main program in *p5.py* does this for you). Here is **run_payroll** (you can just use it):

```
def run_payroll():
    if os.path.exists(PAY_LOGFILE): # pay_log_file is a global variable holding 'payroll.txt'
        os.remove(PAY_LOGFILE)
    for emp in employees:              # employees is the global list of Employee objects
        emp.issue_payment()            # issue_payment calls a method in the classification
                                       # object to compute the pay, which in turn invokes
                                       # the pay method.
```

Every time you issue the payment for an Hourly or Commissioned employee, clear their timecard or receipt lists, respectively, so these entries won't be used again for the next pay period.

Remember that the concrete payment class (**MailMethod** or **DirectMethod**) is responsible for "delivering" the payment by writing to the log file. Therefore, you will want to open the pay log file in "append mode" using 'a' in the second argument to your call to **open** before writing to that file. Make sure the file is closed after appending each new payroll entry.

After you have run the main module in the file *project_e.py* (in Canvas), payroll.txt should contain:

```
Mailing 1911.87 to Karina Gay at 998 Vitae St. Atlanta GA 45169
Transferred 2547.52 for TaShya D. Snow to 465794-3611 at 30417353-K
Mailing 861.20 to Rooney Alvarado at 4963 Nisl. St. Ap #185 Gillette WY 20226
```

Here is a reasonable development sequence:

1. Write **load_employees**. It opens *employees.csv*, ignores the first line, and then reads a line at a time, splitting its arguments on a comma. Create a new Employee object initialized with the string attributes. Then create the appropriate instances for the employee's classification and payment method and bind them as attributes to the new Employee object. Finally, add the Employee object to your global list of employees.
2. Write **find_employee_by_id** by searching the list of employees and returning the Employee object.
3. Implement **Employee**
4. Implement the **Classification** Hierarchy
5. Implement **process_timecards**
6. Implement **process_receipts**
7. Implement the **PayMethod** hierarchy