

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 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, $1/24^{\text{th}}$ of a salary is paid each pay period to employees who receive as 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 $1/24^{\text{th}}$ of their salary.

Commissioned employees also receive $1/24^{\text{th}}$ of their salary along with an additional payment of their total sales times their commission rate.

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. You will do this assignment in two parts:

1. Submit a UML class design document for your classes.
2. Later submit a source file, *payroll.py*, with your implementation.

Requirements (Part 1)

Employee objects have the following required attributes:

- **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 following CRC cards give the information to design your solution:

Employee	
<ul style="list-style-type: none"> • Manage Employee attributes • Change employee's classification • Change employee's payment method • Initiate payment to employee 	<ul style="list-style-type: none"> • Classification • PaymentMethod

Abstract	Classification Hourly, Salaried, Commissioned
<ul style="list-style-type: none"> • Specifies the abstract method issue_payment • Know the employee 	<ul style="list-style-type: none"> • Employee

	Hourly Classification
<ul style="list-style-type: none"> • Know the employee's hourly_rate • Add new time cards • Hold current time cards • Compute the employee's pay • Invoke the pay method 	<ul style="list-style-type: none"> • Employee • PaymentMethod

	Salaried Classification
<ul style="list-style-type: none"> • Know the employee's salary • Invoke the pay method 	<ul style="list-style-type: none"> • Employee • PaymentMethod

Commissioned		Classification
<ul style="list-style-type: none"> • Know the employee's salary, and commission rate • Add receipts • Hold the current sales receipts • Invoke the pay method 	<ul style="list-style-type: none"> • Employee • PaymentMethod 	

Abstract	PaymentMethod	DirectMethod, MailMethod
<ul style="list-style-type: none"> • Specifies the abstract method issue, which posts the employee's payment • Knows the employee 	<ul style="list-style-type: none"> • Employee 	

DirectMethod		PaymentMethod
<ul style="list-style-type: none"> • 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) 	<ul style="list-style-type: none"> • Employee 	

MailMethod		PaymentMethod
<ul style="list-style-type: none"> • Know the employee's name and complete address • Print paycheck (in our case, just prints a line to the log file) 	<ul style="list-style-type: none"> • Employee 	

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 stored in her Hourly classification attribute and the bank routing and account numbers are stored in her DirectMethod paymethod attribute. For TaSha, only the Salary field is stored in her Salaried classification object. All classification and paymethod objects contain a reference to the Employee object so the employee's information is available when payroll is run.

Here is the main program you should use:

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

    # Save copy of payroll file
    shutil.copyfile('paylog.txt', 'paylog1.txt')

    # Change Karina Gay to Salaried and DirectMethod by changing her Employee object:
    emp = find_employee_by_id('688997')
    emp.make_salaried(45884.99)
    emp.direct_method('30417353-K', '465794-3611')

    # Change TaShya Snow to Commissioned and MailMethod; 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)

    # 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)

    # Rerun payroll
    run_payroll()
```

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
...
```

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.