MIS 6382
Object Oriented Programming in Python
Spring 2024
Final Project

The following guidelines should be followed and will be used to grade your final project:

- The code for the final project should be implemented using Jupiter notebooks.
- The solution should be developed using the **Final_Project_Template** Python Notebook file (.ipynb file). This .ipynb file should contain all your inputs and **results/outputs**. The file should be named using your name and the chars "Final". You will be penalized 15% of the grade if your submission does not follow these requirements.
- As long as your code is not too specific and works for the testing examples, it will be considered as right.
- This is **an individual project assignment**; no group submissions will be accepted, and no discussions among classmates are allowed (however, you can google or check the notebooks, videos, etc). Copying solutions from AI tools such as ChatGPT will be marked as 0 points.

Overview of the final project:

- In this final project you will create an employee database application by fulfilling the requirements given below. You must first create the classes as described. Because this final project is largely an extension of homework 4, you can copy your homework 4 code here or use the code provided by the homework 4 solution **when appropriate**. Given that there are more requirements and some different requirements for this final project, please read the whole document carefully and adjust your previous code accordingly.
- After creating the required classes, write a program that will use these classes to build an application for an Employee Database. All employee data is stored in a file called "empdata.dat". This file is provided and will be used to test your program (**don't change this file**). Each time you run the program, the user is allowed to repeatedly select from the seven choices described below:

    1. Add a new employee
    2. Display employees' information
    3. Compute and print the name and compensation of all employees
    4. Search employees by name
    5. Check basic statistics of employees
    6. Calculate the reimbursement of one employee
    7. Exit the application

- The first time you execute the program, your file will have several employees in it. They are used to test your program. As you add employees to the program, it may contain more employees. When you exit the application, all these employees will be stored in **a new data file**. This setting doesn't reflect the real use case, but it fits the final project purpose

(i.e., you will never change the original empdata.dat file so you can debug your program until you get the correct outputs).

- The final project will provide you a detailed program template to follow. What you need to do is to "fill in the blanks" and submit a completed version of the template. This template splits coding task and testing task into small pieces so you don't need to worry about how to complete and test a "huge" program. Just make sure you read the template carefully and follow all the instructions closely. The program template and this pdf document should complement each other; if they contradict somewhere, please let me know.

Requirements of the final project:

First step, create the classes as described.

- Create a base class Employee that has the following attributes:
  - Employee's name (string)
  - Employee's address (string)
  - Vehicle data (Vehicle object).
- The child classes FullTimeEmployee, HourlyEmployee and Consultant that inherit from Employee class have the following additional properties
  - FullTimeEmployee – salary (float).
  - HourlyEmployee - hoursWorked (int) and hourlyRate (float).
  - Consultant – hoursWorked (int) and ProjectType (valid values are 1, 2, and 3).
- The child class Management inherits from both FullTimeEmployee class and Consultant class. Management class has inherited three attributes (i.e. name, address, and vehicle object) indirectly from Employ class, one attribute (i.e. salary) from FullTimeEmployee class and two attributes (i.e. hours worked and project type) from Consultant class.
- All these classes have the __init__ method as well as the **get**, **set, and str** methods. In addition, they have additional methods, i.e. compute_compensation() and compute_reimbursement() as described below. Function compute_compensation() returns weekly compensation and function compute_reimbursement() returns weekly reimbursement.
- Compensation (weekly) for any employee type doesn't call for "outside" information. In other words, all information needed is available after initializing/creating an object. Compensation is to be computed as follows:
  - FullTimeEmployee: Compensation is salary minus taxes and taxes are calculated based on the tax rate in the table below. Please notice that this format calculates the annual

compensation and what this function needs to return is the weekly compensation (assuming there are 52 weeks per year). **This requirement is different from the one in HW4.**

Salary	Tax Rate
$25, 000 or less	18%
> $25,000 and <= $55,000	18% for the first 25,000, 28% for the rest
> $ 55,000	18% for the first 25,000, 28% for the amount between 25,000 and 55,000, and 33% for the rest

For example, someone whose salary is $123,000 will pay 18% on the first 25,000, 28% on the next (55,000 – 25,000) and 33% on the remaining (123,000 – 55,000)

- HourlyEmployee: Compensation is hoursWorked times hourlyRate for the first 40 hours. For hours in excess of 40 hours, the hourly rate is 1.8 times the regular hourly rate. The sum of the two is weekly compensation.
  For example, someone whose hourlyRate is $12.50 and who has worked 48 hours will earn 40 * $12.5 + 8 * $12.5 * 1.8.
- Consultant: Compensation is HourlyRate times the hours worked (this is weekly compensation). HourlyRate for Consultants is computed based on the ProjectType as given in the table below:

Project Type	HourlyRate
1	$55.00
2	$70.00
3	$85.00

- Management: Total weekly compensation is the sum of the compensation from FullTimeEmployee role (i.e. from salary) and the compensation from Consultant role (i.e. from hours worked on a project).

- Reimbursement (weekly) for every employee type calls for “outside” information as input argument(s). In other words, the information is not provided when initializing/creating an object and there is no corresponding attribute. Reimbursement is to be computed as follows:

- FullTimeEmployee: This method will take an argument, which is the annual expense. If the annual expense is no more than $10,000, then the total reimbursement will be equal to the annual expense. If the annual expense is more than $10,000, then the total reimbursement will be equal to $10,000 plus 50 percent of the amount of the annual expense that exceeds $10,000. Please note that this format calculates the annual reimbursement, and what this function needs to return is the weekly reimbursement (assuming there are 52 weeks per year). For example, someone whose annual expense is $12,000 will get $10,000 plus 50% on the remaining ($12,000 – $10,000), totaling $11,000.

Expense	Reimbursement Rate
$10, 000 or less	100%
> $10,000	100% for the first $10,000, 50% for the rest

- HourlyEmployee: This method will take an argument, which is the weekly expense. If the weekly expense is no more than $100, then the total reimbursement will be equal to the weekly expense. If the weekly expense exceeds $100, then the total reimbursement will be equal to $100. For example, someone whose weekly expense is $80 will get a weekly compensation of $80, while someone whose weekly expense is $120 will get a weekly compensation of $100.
- Consultant: This method will take an argument, which is the weekly expense. The weekly reimbursement is the product of weekly expense and the reimbursement rate (which is based on the project type as given in the table below). For example, three consultants all have a weekly expense of $100, then the one working on type 1 project will get $100, the one working on type 2 project will get $90, and the one working on type 3 will get $80.

Project Type	Reimbursement Rate
1	100%
2	90%
3	80%

- Management: This method will take two arguments. The first argument is the annual expense, which fits the FullTimeEmployee role. The second argument is the weekly expense, which fits the Consultant role. Total reimbursement is the sum of the reimbursement from the FullTimeEmployee role (i.e., from the annual expense) and the reimbursement from the Consultant role (i.e., from the weekly expense).
- The Vehicle class is as described below: It has four instance variables – make (string), model (string), year of manufacture (int) and mileage (int). **It should contain get and set methods.** (**This requirement is different from the one in HW4.** It should have a constructor (**__init__**) method which accepts values for all of the instance variables. You should use **aggregation** to include a Vehicle object to your Employee data.

Second step, write a program to take user inputs to work on the Employee Database: (The information here just provides some simple guidelines; see program template for more detailed and specific requirements.)

The Employee Database application contains seven options:

Option 1: Accept input for new Employees. If the user chooses option 1, the program should ask the type of employee first and then guide the user to provide all required information. New employee will be added to a temporary "object container" and written into a new database when program exits.

Option 2: Display employee information. If the user chooses option 2, the program should ask one question (introduced in the program template) and then display employee information.

Option 3: List the names of all employees along with the compensation received by each.

Option 4: Search for employees by name. If the user chooses option 4, the program should ask for a name and then display all employees that match the name search.

Option 5: Display basic statistics (defined in the program template).

Option 6: Calculate the reimbursement of one employee. If the user chooses option 6, the program should ask for a name and let the user select one employee. Then certain information will be asked (depending on employee type) to calculate the weekly reimbursement.

Option 7: Exit the system after writing all information to a new database/file. The program should ask if the user really wants to exit the system before closing the program.

If users select options 1-6, this program should show the option menu again after completing the required job. If users select option 7 and want to exit the system, this program will end then.

Your program should also be able to handle exceptions in some settings. The detailed requirements are provided in the code template. Normally when users enter invalid values, the program should print an appropriate message and ask the user to re-enter a valid value.