HONG KONG INSTITUTE OF VOCATIONAL EDUCATION
Laboratory 3b: Instance and Static Members

TASK:

1. Modify the following program to displays the output shown below:

```
class Vehicle:
    # Important:
    # All variables except constants should be declared as non-public
    MAX_NUM_WHEELS = 16
    MIN_NUM_WHEELS = 2
    DEFAULT_NUM_WHEELS = 4
    __total_vehicle = 0

    def __init__(self, num_wheels):
        self.__total_vehicle += 1
        if (num_wheels < Vehicle.MIN_NUM_WHEELS or
            num_wheels > Vehicle.MAX_NUM_WHEELS):
            self.__num_wheels = Vehicle.DEFAULT_NUM_WHEELS
        else:
            self.__num_wheels = num_wheels

if __name__ == "__main__":
    my_car = Vehicle(4)
    my_truck = Vehicle(Vehicle.MAX_NUM_WHEELS)
    my_bike = Vehicle(Vehicle.MIN_NUM_WHEELS)

    print(f"My car has {my_car.__num_wheels} wheels")
    print(f"My truck has {my_truck.__num_wheels} wheels")
    print(f"My bike has {my_bike.__num_wheels} wheels")
    print(f"There are {Vehicle.__total_vehicle} vehicle(s)")
```

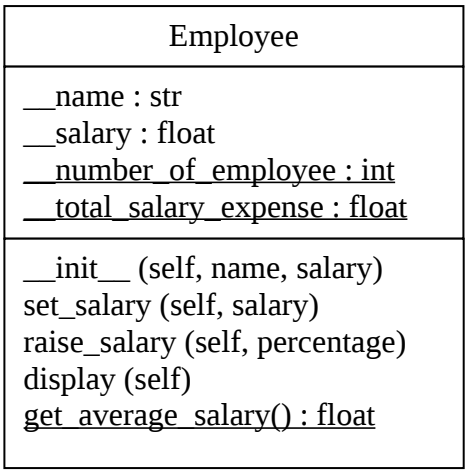
Sample output:
My car has 4 wheels
My truck has 16 wheels
My bike has 2 wheels
There are 3 vehicle(s)
Hints: All variables except constants should be declared as non-public and obtained by public getter methods in client program (data encapsulation rule)

Answer:

```
class Vehicle:
    # Important:
    # All variables except constants should be declared as non-public
    MAX_NUM_WHEELS = 16
    MIN_NUM_WHEELS = 2
    DEFAULT_NUM_WHEELS = 4
    __total_vehicle = 0

    def __init__(self, num_wheels):
        Vehicle.__total_vehicle += 1
        if (num_wheels < Vehicle.MIN_NUM_WHEELS or
            num_wheels > Vehicle.MAX_NUM_WHEELS):
            self.__num_wheels = Vehicle.DEFAULT_NUM_WHEELS
        else:
            self.__num_wheels = num_wheels

    def get_num_wheel(self):
        return self.__num_wheels

    @staticmethod
    def get_total_vehicle():
        return Vehicle.__total_vehicle

if __name__ == "__main__":
    my_car = Vehicle(4)
    my_truck = Vehicle(Vehicle.MAX_NUM_WHEELS)
    my_bike = Vehicle(Vehicle.MIN_NUM_WHEELS)

    print(f"My car has {my_car.get_num_wheel()} wheels")
    print(f"My truck has {my_truck.get_num_wheel()} wheels")
    print(f"My bike has {my_bike.get_num_wheel()} wheels")
    print(f"There are {Vehicle.get_total_vehicle()} vehicle(s)")
```

2. Write a supplier class and client / driver program in the following parts below:

(a) Create a class Employee that has:



Attributes:

__name – to store the name of the employee
__salary – to store the current salary of the employee
__number_of_employee – a static variable to store the total number of employees created
__total_salary_expense – a static variable to store the total salary expense of all employees

Methods:

__init__ – initialize the name and salary Attribute in employee object. Add 1 to Number of employee and add salary to Total salary expenses when employee object is created
set_salary – if the salary being update is greater than 0, minus the current salary from total_salary_expense first, update the current salary of employee object's and then add it to total_salary_expense
raise_salary – to raise the salary of the employee by percentage given. Note: you have to perform all the operations in the set_salary method, such as checking for salary > 0, minus the current salary from total_salary_expense, update the raised salary and add it to total_salary_expense
display – to print the output with the sample given
get_average_salary – a static method to return the average salary of all employees

Answer:

```
class Employee:
    __number_of_employee = 0
    __total_salary_expense = 0
    def __init__(self, name, salary):
        self.__name = name
        self.__salary = salary
        Employee.__number_of_employee += 1
        Employee.__total_salary_expense +=salary
    def set_salary(self, salary):
        if salary > 0:
            Employee.__total_salary_expense -= self.__salary
            self.__salary = salary
            Employee.__total_salary_expense += salary
    def raise_salary(self, percentage):
        self.set_salary(self.__salary * (1+percentage/100))
    def display(self):
        print (f"Employee name={self.__name}, salary={self.__salary:.0f}")
    @staticmethod
    def get_average_salary():
        return Employee.__total_salary_expense / Employee.__number_of_employee
```

(b) Complete the following client / driver program that will do the following:

(i) Create a **list** named **emp_list** to hold all employee objects

(ii) Create an **employee object** with name **"Chan Tai Man"** and salary **12000**. **Add** it to the list you created in step 1

(iii) Create an **employee object** with name **"Tam Ping Shing"** and salary **13500**. **Add** it to the list you created in step 1

(iv) Create an **employee object** with name **"Leung Pig Hung"** and salary **15000**. **Add** it to the list you created in step 1

(v) Print the **name** and **salary** of the **three employee objects** as shown in the sample output by calling the **employee display method**.

(vi) Print the **average salary** of **employee objects** created as shown in the sample output by calling the **static method get_average_salary**.

(vii) Raise the salary of employee "Chan Tai Man" by 10%.

(viii) Raise the salary of employee "Tam Ping Shing" by 5%

(ix) Set salary of employee "Leung Pig Hung" to 9000

(x) Print the employee information and average salary again as shown in the sample output.

Sample output:
Before:
Employee name=Chan Tai Man, salary=12000
Employee name=Tam Ping Shing, salary=13500
Employee name=Leung Pig Hung, salary=15000
Average salary is 13500.0
After:
Employee name=Chan Tai Man, salary=13200
Employee name=Tam Ping Shing, salary=14175
Employee name=Leung Pig Hung, salary=9000
Average salary is 12125.0

```
if __name__ == "__main__":
    # 1 Create a list named emp_list to hold all
    #   employee objects


    # 2 - 4 Create employee object with name and salary
    #       Add it to the list you created in step 1


    print ("Before:")
    # 5 Use looping and call display method to print
    #   employee information


    # 6 Print average salary of employee objects


    print ("After:")
    # 7 Raise salary of employee "Chan Tai Man" by 10%


    # 8 Raise salary of employee "Tam Ping Shing" by 5%


    # 9 Set salary of employee "Leung Pig Hung" to 9000


    # 10 Print employee information and average salary again
```

Answer:

```
if __name__ == "__main__":
    # 1 Create a list named emp_list to hold all employee objects
    emp_list = list()

    # 2 - 4 Create employee object with name and salary
    #       Add it to the list you created in step 1
    emp_list.append(Employee("Chan Tai Man", 12000))
    emp_list.append(Employee("Tam Ping Shing", 13500))
    emp_list.append(Employee("Leung Pig Hung", 15000))
    print ("Before:")

    # 5 Use looping and call display method to print
    #   employee information
    for emp in emp_list:
        emp.display()

    # 6 Print average salary of employee objects
    print(f"Average salary is {Employee.get_average_salary()}")

    print ("After:")
    # 7 Raise salary of employee "Chan Tai Man" by 10%
    emp_list[0].raise_salary(10)

    # 8 Raise salary of employee "Tam Ping Shing" by 5%
    emp_list[1].raise_salary(5)

    # 9 Set salary of employee "Leung Pig Hung" to 9000
    emp_list[2].set_salary(9000)

    # 10 Print employee information and average salary again
    for emp in emp_list:
        emp.display()
    print(f"Average salary is {Employee.get_average_salary()}")
```