**PMT: Hotel Employees**

A three-star hotel will be opening up in the Pacific Islands next Spring. The owners are trying to determine what their monthly expenditure will be for employee salaries / wages. They plan to have three categories of employees: **managers**, **supervisors**, and **entry-level** staff. The **monthly** wages for these categories of employees are determined as follows:

**Managers**: A basic salary of $6,000, an allowance of $1,000 for medical expenses, a wardrobe allowance of $500, and an education allowance of $1,000. An **annual** bonus of $3,600 is also budgeted for managers.

**Supervisors**: A basic salary of $3,000, $800 for medical expenses, a wardrobe allowance of $200, and an education allowance of $500. Supervisors will also receive an **annual** bonus of $2,400.

**EntryLevel**: A basic salary of $1,500, $500 for medical expenses, and an education allowance of $400. Entry-level employees will also receive an **annual** bonus of $1,200.

Write a Python application to assist the owners of the hotel to compute their **monthly** expenditure on employees’ wages. Your application should have the following classes:

* **Employee**
* Classes **Manager**, **Supervisor**, and **EntryLevel** which are subclasses of **Employee**
* Class **HotelApp**
* For each class, write initializer methods and determine what parameters (if any) should be passed to them.
* Also consider what attributes would be needed in the initializer methods and include them as appropriate.
* Also consider if you need to override any of the methods in the Employee class in the subclasses.

**Note**: You should not make any changes to the method names or parameter lists for the methods shown for each class.

The test function **main()** is also provided and **should not be modified**.

The methods in class **HotelApp** are shown below:

**class** HotelApp:  
  
 *# Return a list of the hotel's employees  
 # @return A list of the hotel's employees* **def** get\_employees(self):  
  
 *# Add the employee object to the list of employees of the hotel.  
 # @param employee: The employee to be added* **def** add\_employee(self, employee):  
  
 *# Computes and returns the total monthly wages of all the  
 # employees of the hotel.  
 # @return Total wages of all employees* **def** get\_monthly\_employee\_wages(self):

The methods in the Employee class are shown below:

**class** Employee:  
  
 *# Return this employee's name.  
 # @return This employee's name.* **def** get\_name(self):  
  
 *# Calculate and return the monthly wages for this employee.  
 # @return This employee's monthly wages* **def** get\_monthly\_wage(self):

Also, you should test your classes using the test function below:

**def** main():  
 hotel = HotelApp()  
 emp1 = Manager(**"John Smith"**)  
 emp2 = Supervisor(**"Jane Jones"**)  
 emp3 = EntryLevel(**"Ruth Sharp"**)  
  
 hotel.add\_employee(emp1)  
 hotel.add\_employee(emp2)  
 hotel.add\_employee(emp3)  
  
 employees = hotel.get\_employees()  
  
 print(**"Employee List"**)  
 print(**"=============\n"**)  
  
 **for** i **in** range(len(employees)):  
 print(**f'{**employees[i].get\_name()**:<17}{"$"}{**employees[i].get\_monthly\_wage()**:>12,.2f}'**)  
  
 print()  
 print(**f'{"Total wages: $"}{**hotel.get\_monthly\_employee\_wages()**:>10,.2f}'**)  
  
 emp4 = Manager(**"Dempsey Dean"**)  
 emp5 = EntryLevel(**"Sophia Weather"**)  
  
 hotel.add\_employee(emp4)  
 hotel.add\_employee(emp5)  
  
 print()  
 **for** i **in** range(len(employees)):  
 print(**f'{**employees[i].get\_name()**:<17}{"$"}{**employees[i].get\_monthly\_wage()**:>12,.2f}'**)  
  
 print()  
 print(**f'{"Total wages: $"}{**hotel.get\_monthly\_employee\_wages()**:>10,.2f}'**)

**Expected output:**

Employee List

=============

John Smith $ 8,800.00

Jane Jones $ 4,700.00

Ruth Sharp $ 2,500.00

Total wages: $ 16,000.00

John Smith $ 8,800.00

Jane Jones $ 4,700.00

Ruth Sharp $ 2,500.00

Dempsey Dean $ 8,800.00

Sophia Weather $ 2,500.00

Total wages: $ 27,300.00

Process finished with exit code 0

**PMT Submission**

Submit your completed program to the dropbox before it closes.