**Creating a Simple Payroll System.**

The HR system needs to process payroll for the company's employees, but there are different types of employees depending on how their payroll is calculated.

1. implement a base class Employee that handles the common interface for every employee type. Employee is the base for all employee types. It is constructed with an emp\_id and a name i.e every Employee must have an id assigned as well as a name. The HR system requires that every Employee processed must have a calculated\_payroll() method that returns weekly salary for the employee.The implementation of that interface differs depending on the type of Employee.
2. Create a derived class called SalaryEmployee that inherits Employee. The class is initialized with emp\_id and name required by the base class, use super() to inherit the members of the base class. SalaryEmployee also requires a weekly\_salary initialization parameter that represents the amount the employee makes per week. The class provides the reauired calculate\_payroll() method used by the HR system. The implementation just returns the amount stored in weekly\_salary.
3. The company also employs manufacturing workers that are paid by the hour, so you add an HourlyEmployee to the HR system. The HourlyEmployee class is initialized with emp\_id and name, like the base class, plus the hours\_worked and the hourly\_rate required to calculate the payroll. The calculate\_payroll() method is implemented by returning thr hours worked times the hour rate.
4. Finally, the company employs sales associates that are paid through a fixed salary plus a commission based on their sales, so you create a CommissionEmployee class. Create a class called CommissionEmployee that inherits from SalaryEmployee because both classes have a weekly\_salary to consider. At the same time, CommissionEmployee is initialized with a commission value that is based on the sales for the employee .The calculate\_payroll() leverages the implementation of the base class to retrieve the fixed salary and adds the commission value.

Since CommissionEmployee derives from SalaryEmployee, you have access to the weekly\_salary property directly and you could've implemented calculate\_payroll() using the value of that property.

Write a python program to implement the above

**ANSWER**

from abc import ABC, abstractmethod

class Employee(ABC):

def \_\_init\_\_(self, emp\_id: int, name: str):

self.emp\_id = emp\_id

self.name = name

def calculate\_payroll(self) -> float:

"""

Calculate and return the weekly salary for the employee.

Must be implemented by all subclasses.

"""

pass

class SalaryEmployee(Employee):

def \_\_init\_\_(self, emp\_id: int, name: str, weekly\_salary: float):

super().\_\_init\_\_(emp\_id, name)

self.weekly\_salary = weekly\_salary

def calculate\_payroll(self) -> float:

return self.weekly\_salary

class HourlyEmployee(Employee):

def \_\_init\_\_(self, emp\_id: int, name: str, hours\_worked: float, hourly\_rate: float):

super().\_\_init\_\_(emp\_id, name)

self.hours\_worked = hours\_worked

self.hourly\_rate = hourly\_rate

def calculate\_payroll(self) -> float:

return self.hours\_worked \* self.hourly\_rate

class CommissionEmployee(SalaryEmployee):

def \_\_init\_\_(self, emp\_id: int, name: str, weekly\_salary: float, commission: float):

super().\_\_init\_\_(emp\_id, name, weekly\_salary)

self.commission = commission

def calculate\_payroll(self) -> float:

base\_pay = super().calculate\_payroll() # base weekly salary

return base\_pay + self.commission

if \_\_name\_\_ == "\_\_main\_\_":

employees = [

SalaryEmployee(1, "Alice", weekly\_salary=1200.0),

HourlyEmployee(2, "Bob", hours\_worked=40, hourly\_rate=20.0),

CommissionEmployee(3, "Charlie", weekly\_salary=800.0, commission=200.0)

]

for emp in employees:

print(f"Employee: {emp.name}, Payroll: ${emp.calculate\_payroll():.2f}")