Laboratory Activity 3: Inheritance, Encapsulation, and Abstraction

Create a program in python that satisfies the following:

Inheritance, Encapsulation, and Abstraction concept with ADT list

Class(Employee: emp_id. emp_name, emp_address, Fulltime: allowance, rate, PartTime: rate)

Class(Salary: salary_id, Salary, cut_off_date, days_of_work)

INPUT:

```
# Base Employee class
class Employee:
    def __init__(self, emp_id, emp_name, emp_address):
        self.emp_id = emp_id
        self.emp_name = emp_name
        self.emp_address = emp_address

    def calculate_salary(self, days_of_work):
        pass

    # Adding a method to return employee details
    def get_employee_details(self):
        return f"ID: {self.emp_id}, Name: {self.emp_name}, Address:
{self.emp_address}"

# FullTime Employee class manually calling Employee constructor
class FullTime(Employee):
    def __init__(self, emp_id, emp_name, emp_address, allowance, rate):
        Employee.__init__(self, emp_id, emp_name, emp_address)
        self.allowance = allowance
        self.rate = rate

    def calculate_salary(self, days_of_work):
        return self.allowance + (self.rate * days_of_work)

# PartTime Employee class manually calling Employee constructor
class PartTime(Employee):
    def __init__(self, emp_id, emp_name, emp_address, rate):
        Employee.__init__(self, emp_id, emp_name, emp_address)
        self.rate = rate
```

```
    def calculate_salary(self, days_of_work):
        return self.rate * days_of_work

# Salary class to handle salary calculation for an employee, with
cut_off_date as a string
class Salary:
    def __init__(self, salary_id, employee, days_of_work, cut_off_date):
        self.salary_id = salary_id
        self.employee = employee
        self.days_of_work = days_of_work
        # Store the cut_off_date as a simple string
        self.cut_off_date = cut_off_date

    def calculate_total_salary(self):
        return self.employee.calculate_salary(self.days_of_work)

    def get_salary_details(self):
        return f"Salary ID: {self.salary_id}, Employee:
{self.employee.emp_name}, Cut-off Date: {self.cut_off_date}, Total Salary:
{self.calculate_total_salary()}"

# Example usage
if __name__ == "__main__":

    # Full-time employee example (Rigor Batumbakal)
    full_time_emp = FullTime(emp_id=1294273, emp_name="Rigor Batumbakal",
emp_address="18 Kalayaan St.", allowance=1000, rate=700)

    # Part-time employee example (Pedro Penduko)
    part_time_emp = PartTime(emp_id=876212, emp_name="Pedro Penduko",
emp_address="34 Maligaya St.", rate=350)

    # Salary calculations for a full-time employee with a cut-off date
    full_time_salary = Salary(salary_id=7623, employee=full_time_emp,
days_of_work=24, cut_off_date="2024-09-30")

    # Print full-time employee details and salary
    print(full_time_emp.get_employee_details())
    print(full_time_salary.get_salary_details())
```

```
    # Salary calculations for a part-time employee with a cut-off date
    part_time_salary = Salary(salary_id=121414, employee=part_time_emp,
days_of_work=16, cut_off_date="2024-09-30")

    # Print part-time employee details and salary
    print(part_time_emp.get_employee_details())
    print(part_time_salary.get_salary_details())
```

OUPUT:

```
ID: 1294273, Name: Rigor Batumbakal, Address: 18 Kalayaan St.
Salary ID: 7623, Employee: Rigor Batumbakal, Cut-off Date: 2024-09-30, Total Salary: 17800
ID: 876212, Name: Pedro Penduko, Address: 34 Maligaya St.
Salary ID: 121414, Employee: Pedro Penduko, Cut-off Date: 2024-09-30, Total Salary: 5600
```