In [3]:
```
#Employee Salary Management (Abstraction)
#You are tasked with building an employee salary management system. Use abstract
#● Abstract Class: Employee with abstract methods calculate_salary() andget_empl
#● FullTimeEmployee: Overrides calculate_salary() by considering a monthly fixed
#● PartTimeEmployee: Overrides calculate_salary() by considering an hourly rate
#Task:
#1. Create the abstract class and its subclasses.
#2. Implement the salary calculation for both types of employees.
#3. Instantiate both employee types, calculate salaries, and display their detai
#4. Add an abstract method raise_salary() that forces both subclasses to impleme

from abc import ABC, abstractmethod

class Employee(ABC):
    def __init__(self, name, employee_id):
        self.name = name
        self.employee_id = employee_id

    @abstractmethod
    def calculate_salary(self):
        pass

    @abstractmethod
    def get_employee_details(self):
        pass

    @abstractmethod
    def raise_salary(self, amount):
        pass


class FullTimeEmployee(Employee):
    def __init__(self, name, employee_id, monthly_salary):
        super().__init__(name, employee_id)
        self.monthly_salary = monthly_salary

    def calculate_salary(self):
        return self.monthly_salary

    def get_employee_details(self):
        return f"Full-Time Employee: {self.name}, ID: {self.employee_id}, Monthl

    def raise_salary(self, amount):
        self.monthly_salary += amount
        return f"Salary raised by ${amount:.2f}. New monthly salary: ${self.mont


class PartTimeEmployee(Employee):
    def __init__(self, name, employee_id, hourly_rate, hours_worked):
        super().__init__(name, employee_id)
        self.hourly_rate = hourly_rate
        self.hours_worked = hours_worked

    def calculate_salary(self):
        return self.hourly_rate * self.hours_worked

    def get_employee_details(self):
        return f"Part-Time Employee: {self.name}, ID: {self.employee_id}, Hourly
```

```
    def raise_salary(self, amount):
        self.hourly_rate += amount
        return f"Hourly rate raised by ${amount:.2f}. New hourly rate: ${self.ho

if __name__ == "__main__":
    full_time = FullTimeEmployee("John Doe", "FT001", 5000)
    part_time = PartTimeEmployee("Jane Smith", "PT001", 20, 80)

    print(full_time.get_employee_details())
    print(f"Calculated Salary: ${full_time.calculate_salary():.2f}\n")

    print(part_time.get_employee_details())
    print(f"Calculated Salary: ${part_time.calculate_salary():.2f}\n")

    print(full_time.raise_salary(500))
    print(f"New Calculated Salary: ${full_time.calculate_salary():.2f}\n")

    print(part_time.raise_salary(2.5))
    print(f"New Calculated Salary: ${part_time.calculate_salary():.2f}")
```

```
Full-Time Employee: John Doe, ID: FT001, Monthly Salary: $5000.00
Calculated Salary: $5000.00

Part-Time Employee: Jane Smith, ID: PT001, Hourly Rate: $20.00, Hours Worked: 80
Calculated Salary: $1600.00

Salary raised by $500.00. New monthly salary: $5500.00
New Calculated Salary: $5500.00

Hourly rate raised by $2.50. New hourly rate: $22.50
New Calculated Salary: $1800.00
```

In [ ]: