Решение тестовой задачи (Анастасия Радцевич)

Компанию описывает класс Company. У компании есть название - свойство Name типа string, сотрудники - приватное свойство Employees типа List<Person>. Можно добавить сотрудника в компанию – метод AddEmployee, а также рассчитать суммарную зарплату всех сотрудников компании – метод AllSalaryCalculation.

Так как из условия понятно, что сотрудники должны быть реализованы в виде иерархической древовидной структуры я применила паттерн Composite.

Общий тип сотрудника описывает абстрактный класс Person. Он характерен для всех компонентов в древовидной структуре. У сотрудника есть имя – свойство Name типа string, дата поступления на работу – свойство AddOnDate типа DateTime, базовая ставка – свойство BaseRate типа decimal, начальник – свойство Head типа PersonWithSubordinates. По условию базовая ставка одинакова для всех сотрудников, поэтому значение свойству присваивается в конструкторе класса Person. В дальнейшем можно хранить ставки сотрудников разных типов в перечислении. Можно добавить начальника сотруднику – метод AddHead. Можно рассчитать зарплату сотрудника – метод SalaryCalculation. Можно получить зарплату всех подчиненных – метод GetSubordinatesSalary.

Абстрактный класс PersonWithSubordinates описывает сотрудника, у которого могут быть подчиненные, он наследуется от класса Person. У него есть подчиненные – свойство Subordinates типа List<Peson>. Можно добавить подчиненного – метод AddSubordinates. Также он реализует механизм подсчета зарплаты всех подчиненных – метод GetSubordinatesSalary переопределенный от Person.

Класс Employee описывает сотрудника, который не может иметь подчиненных. Наследуется от Person и реализует механизм подсчета зарплаты сотрудника этого типа – метод SalaryCalculation переопределенный от Person.

Классы Manager и Sales описывают сотрудников, у которых могут быть подчиненные. Наследуются от PersonWithSubordinates и реализуют механизм подсчета зарплаты сотрудников этого типа – метод SalaryCalculation переопределенный от Person.

Можно было реализовать паттерн стратегия для расчета зарплаты сотрудника, в зависимости от типа. И тогда можно было не делать классы Employee, Manager и Sales. Но в дальнейшем может появиться необходимость добавить что-то специфичное для определенного типа сотрудника, поэтому мое решение более расширяемое.

В реальности нужно будет хранить данные о сотрудниках компании. Поэтому можно будет применить паттерн стратегия для реализации метода AddEmployee класса Company, что позволит использовать нужную реализацию.