Test task solution (Anastasiya Radtsevich)

The company is described by the Company class. The company has a name – the Name property of the string type, employees – the private Employees property of the List<Person> type. You can add an employee to the company – the AddEmployee method, and also calculate the total salary of all employees in the company – the AllSalaryCalculation method.

As it is understandable from the condition, employees should be implemented in the form of a hierarchical tree structure, I applied the Composite pattern.

The general employee type describes the abstract class Person. It is common to all components in the tree structure. The employee has a name – the Name property of the string type, the day of employment – the AddOnDate property of the DateTime type, the base rate – the BaseRate property of the decimal type, the head – the Head property of the PersonWithSubordinates type. The base rate is the same for all employees, by the condition, so the property value is assigned in the constructor of the Person class. It is possible to store the rates for different types of employee in the enum type object. You can add a head to an employee – the AddHead method. You can calculate the salary of an employee – the SalaryCalculation method. You can get the salaries of all subordinates – the GetSubordinatesSalary method.

The abstract class PersonWithSubordinates describes an employee who can have subordinates, it is inherited from the Person class. It has subordinates – the Subordinates property of the List<Person> type. You can add a subordinate – the AddSubordinates method. It also implements the salary calculating mechanism for all subordinates – the GetSubordinatesSalary method overridden from Person class.

The Employee class describes an employee who can not have subordinates. It is inherited from Person class and it implements the salary calculating mechanism for this type of employee – the SalaryCalculation method overridden from Person class.

The Manager and Sales classes describe an employee who can have subordinates. They are inherited from PersonWithSubordinates and they implement the salary calculating mechanism for this type of employees – the SalaryCalculation method overridden from Person class.

I could implement the strategy pattern for employee salary calculating. And I could not make the Employee, Manager and Sales classes. My solution is more extensible because of possibility to add something for a specific type of employee.

In reality, you have to store the company’s employees data somewhere. So it is possible to apply the strategy pattern for implement to the AddEmployee method of the Company class.