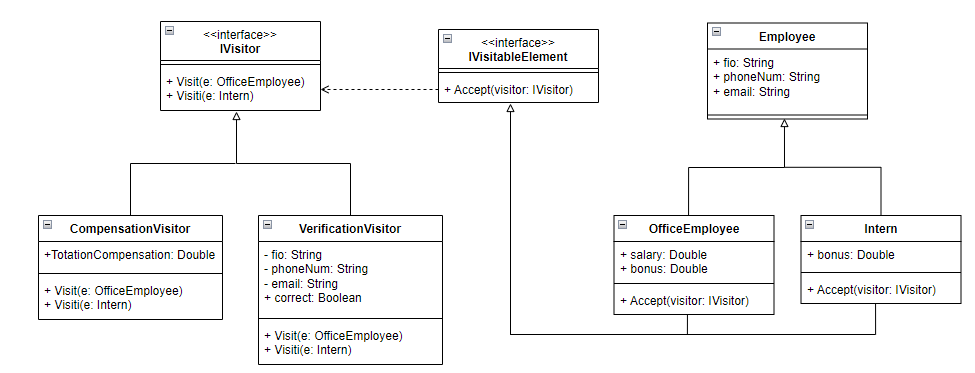
Диаграмма классов:



Паттерн применяется при добавлении нового сотрудника для проверки корректности заполнения данных (почта записана как почта; ФИО – это фамилия и инициалы; номер телефона имеет вид +7-111-222-3333 или 8-111-222-3333; зарплата не может быть меньше или равна 0), а также для подсчета общего расхода на оклад сотрудников (бонус + оклад у работников офиса, и просто бонус у стажёров).

Паттерн упрощает работу с классами сотрудников, т. к. всю необходимую работу с полями выполняют методы Visit у классов посетителя, которые вызываются в методе Accept у классов сотрудников. Это помогает избежать изменения каждого класса сотрудника по отдельности для определения действий над ними, а также избавляет от проблем с изменением кода при добавлении нового класса сотрудника или нового действия нал существующими.

Вся работа с классами сотрудников сводится к вызову метода Accept, с необходимым посетителем в качестве параметра, для выполнения определённых действий:

OfficeEmployee employee = new OfficeEmployee(fio, phoneNum, email, salary, bonus);

var verificationVisitor = new VerificationVisitor();

employee.Accept(verificationVisitor);

Или

var compenstionVisitor = new CompensationVisitor();

mainform.employees.ForEach(em => em.Accept(compenstionVisitor));

Без паттерна всю реализацию метода Accept пришлось бы определять напрямую в коде или создавать отдельные методы внутри классов сотрудников, причём для каждого класса реализация была бы своя:

// Класс сотрудника

public class OfficeEmployee : Employee

{

public double salary { get; set; }

public double bonus { get; set; }

public bool correctFio = true;

public bool correctPhoneNum = true;

public bool correctEmail = true;

public bool correctSalary = true;

private string e = @"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\\*\+/=\?\^`\{\}\|~\w])\*)(?<=[0-9a-z])@))" +

@"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]\*[0-9a-z]\*\.)+[a-z0-9]{2,17}))$";

private string p = @"((\+[0-9])|[0-9])-[0-9]{3}-[0-9]{3}-[0-9]{4}$";

private string f = @"\D+\s\D\.\s\D\.";

public OfficeEmployee(string fio, string phoneNum, string email, double salary, double bonus) : base(fio, phoneNum, email)

{

this.salary = salary;

this.bonus = bonus;

}

public void Check(string email, string phoneNum, string fio, double salary)

{

if (!Regex.IsMatch(email, e, RegexOptions.IgnoreCase)) correctEmail = false;

if (!Regex.IsMatch(phoneNum, p, RegexOptions.IgnoreCase)) correctPhoneNum = false;

if (!Regex.IsMatch(fio, f, RegexOptions.IgnoreCase)) correctFio = false;

if (salary <= 0) correctSalary = false;

}

}

// Непосредственно при добавлении нового сотрудника

private void addButton\_Click(object sender, EventArgs e)

{

string fio = fioTextBox.Text;

string phoneNum = phoneNumTextBox.Text;

string email = emailTextBox.Text;

double salary = Convert.ToDouble(salaryTextBox.Text);

double bonus = Convert.ToDouble(bonusTextBox.Text);

OfficeEmployee employee;

employee.Check(email, phoneNum, fio, salary);

if (!employee.correctFio)

{

ErrorForm errorForm = new ErrorForm();

errorForm.errorLabel.Text = "Поле ФИО должно быть заполненно в виде Иванов И. И.";

errorForm.Show();

}

else if (!employee.correctPhoneNum)

{

ErrorForm errorForm = new ErrorForm();

errorForm.errorLabel.Text = "Поле номер телефона должно быть заполнено в виде +7-111-222-3333\r\nИли 8-111-222-3333\r\n";

errorForm.Show();

}

else if (!employee.correctEmail)

{

ErrorForm errorForm = new ErrorForm();

errorForm.errorLabel.Text = "Поле Email должно быть заполнено в виде ex.mail@mail.ru";

errorForm.Show();

}

else if (!employee.correctSalary)

{

ErrorForm errorForm = new ErrorForm();

errorForm.errorLabel.Text = "Зарплата сотрудника не может быть меньше или равна 0";

errorForm.Show();

}

else

{

mainform.compensationLabel.Text = Convert.ToString(compenstionVisitor.TotationCompensation);

}

}