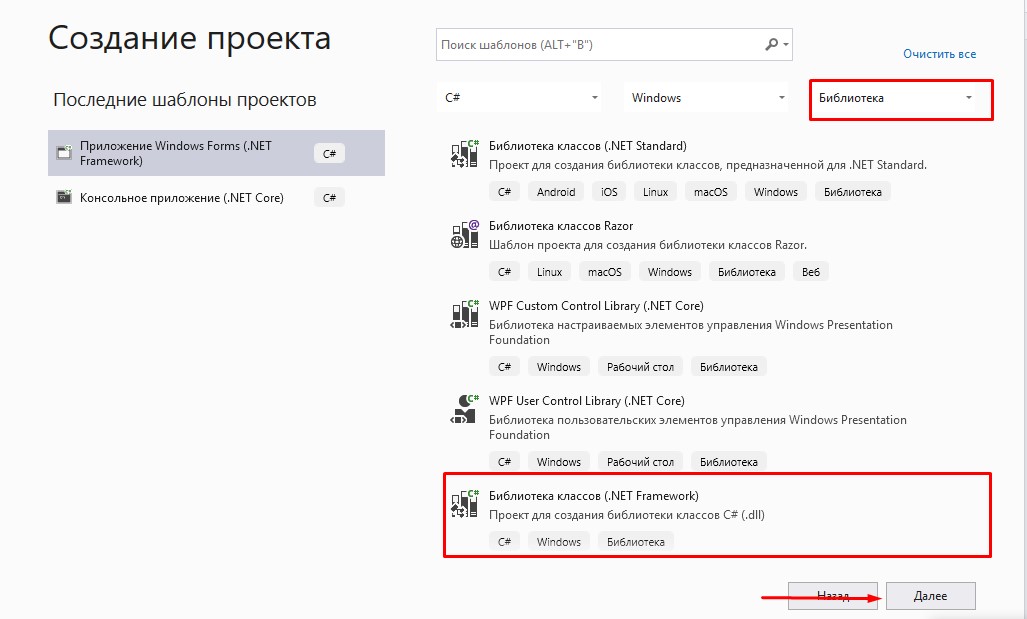
**Работа с Unit-тестами**

Задание 1. Простой Unit-тест

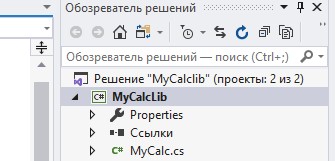
Порядок выполнения работы:

В MS Visual Studio создайте проект, выбрав тип проекта – библиотека



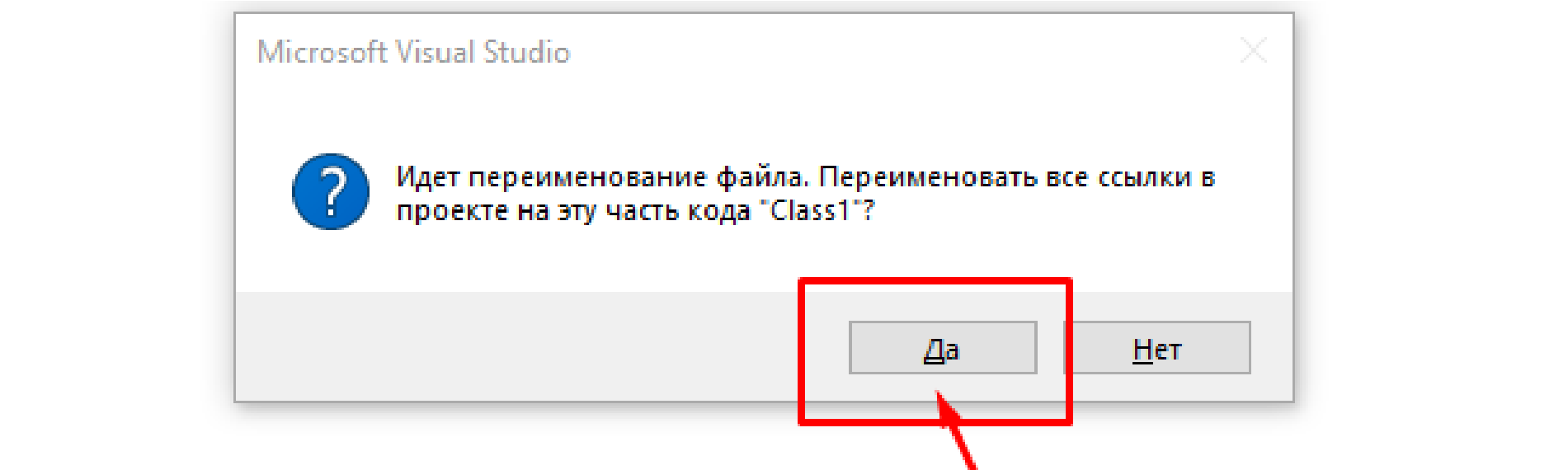
Назовите проект MyCalcLib.

При создании проекта автоматически генерируется класс и создается решение и проект (если при создании проекта вы указали создать проект и решение в одной директории)

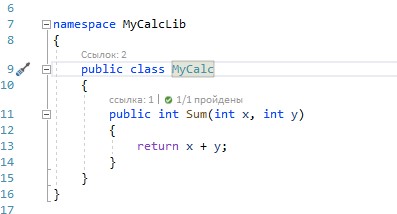


Переименуйте Class1 в MyCalc, для этого в обозревателе решений выделите название класса и через контекстное меню выберите команду «Переименовать», или нажмите F2.

При переименовании обязательно подтвердите изменение всех ссылок на объект в проекте:

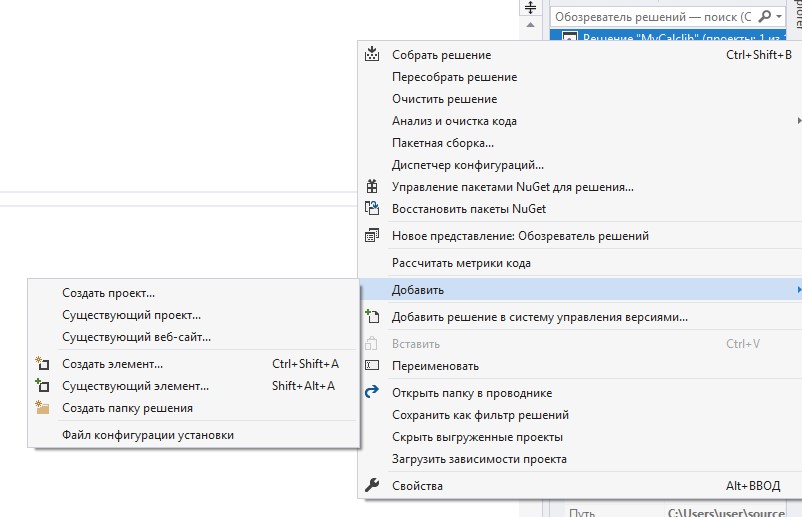


Далее в коде добавим открытый метод Sum, которому на входе передаем два параметра и который возвращает сумму этих параметров:

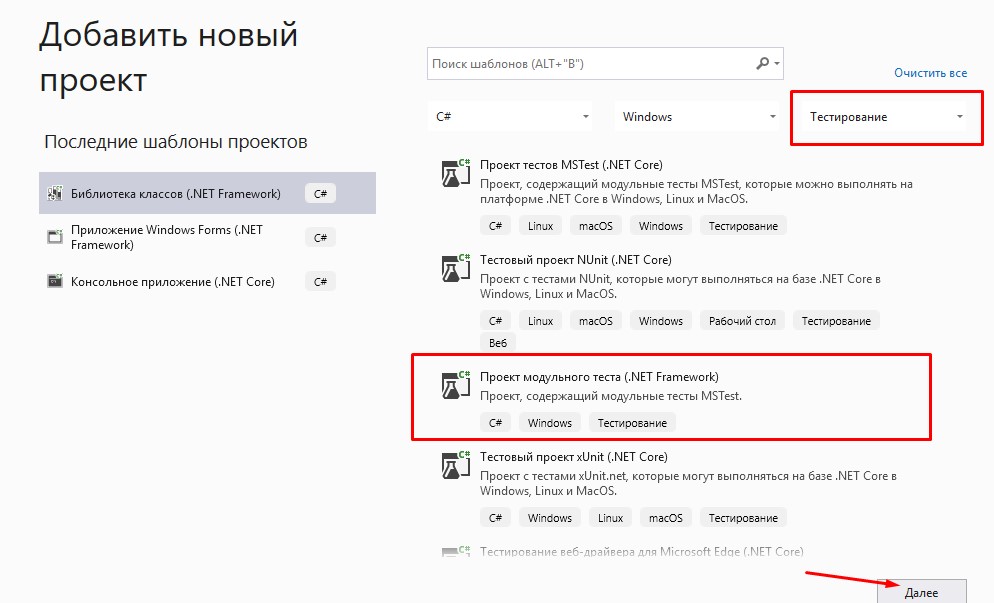


Проект, который мы написали – библиотека, юнит-тест мы должны вынести в отдельный проект.

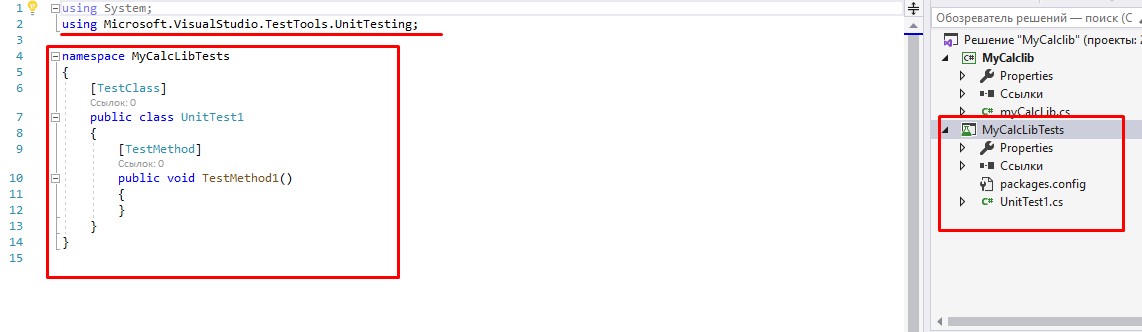
Для этого в решении MyCalcLib добавим unit-тест (Добавить – Создать проект):



Для этого в разделе «Тестирование» выберем тип проекта – модульный тест:

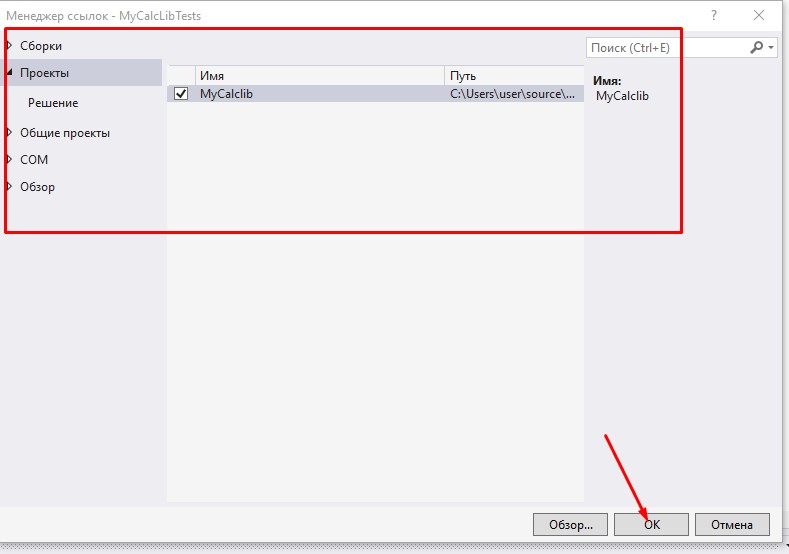


Назовем проект MyCalcLibTests. По умолчанию IDE генерирует код и создает проект:

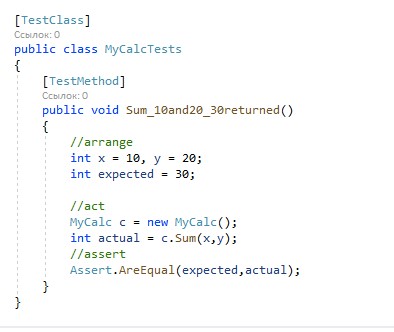


Обратите внимание, что в созданном юнит тесте добавлено пространство имен для тестирования.

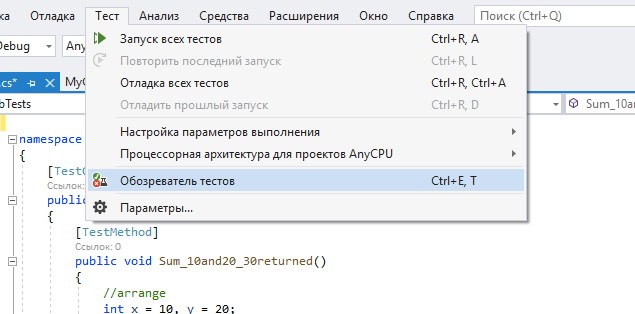
Для того, чтобы модульный тест можно было использовать для тестирования библиотеки – необходимо подключить сборку. Для этого в обозревателе решений в ссылках подключим проект из существующего решения:



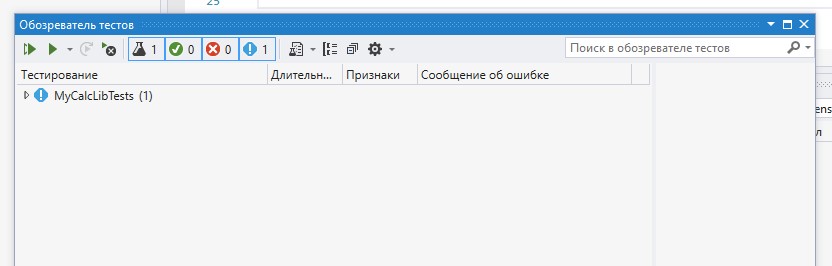
Переименуйте Class1 в MyCalcLibTests и создайте метод по шаблону ААА:



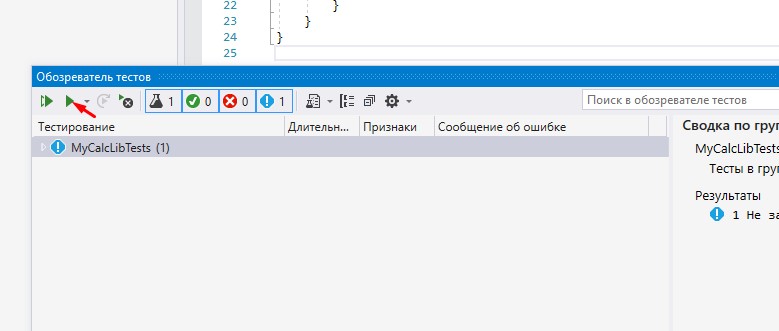
Откроем окно просмотра тестов:



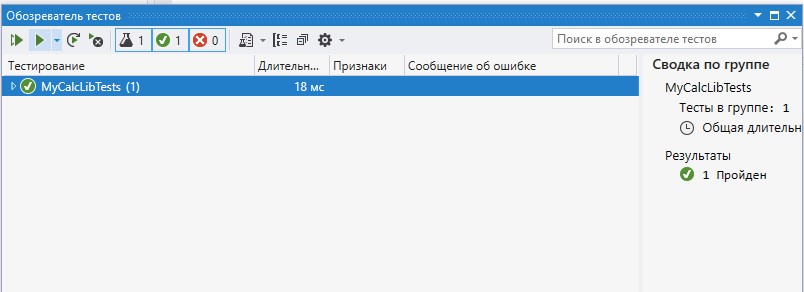
Видим список всех тестов



Для запуска теста:



Тест выполнен верно:



Задание 2.

Создайте проект для проверки пароля при регистрации пользователя. Именования всех компонентов сформируйте самостоятельно.

Код проекта:

using System;

using System.Text.RegularExpressions;

namespace UserRegistration

{

public static class PasswordStrengthCheker

{

// Возвращает значение определяющее сложность пароля пользователя.

public static int GetPasswordStrength(string password)

{

if (string.IsNullOrEmpty(password))

{

return 0;

}

int result = 0;

// +1 балл за длину.

if (Math.Max(password.Length, 7) > 7)

{

result++;

}

//+1 балл за наличие символа в нижнем регистре if (Regex.Match(password, "[a-z]").Success)

{

result++;

}

//+1 балл за наличие символа в верхнем регистре if (Regex.Match(password, "[A-Z]").Success)

{

result++;

}

// +1 балл за наличие числа.

if (Regex.Match(password, "[0-9]").Success)

{

result++;

}

// +1 балл за наличие специального символа.

if (Regex.Match(password,

"[\\!\\@\\#\\$\\%\\^\\&\\\*\\(\\)\\{\\}\\[\\]\\:\\'\\;\\\"\\/\\?\\.\\>\\,\\<\\~\\`\\-\\\\\_\\=\\+\\|]").Success)

{

result++;

}

return result;

}

}

}

Реализуйте юнит-тест в этом же решении отдельным проектом, самостоятельно определив именования:

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace UserRegistration.Tests

{

[TestClass()]

public class PasswordStrengthChekerTests

{

[TestMethod]

public void GetPasswordStrength\_AllCahrs\_5Points()

{

// arrange

string password = "P2ssw0rd#"; int expected = 5;

// act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_UpperCase\_3Points()

{

// Arrange

string password = "Password";

int expected = 3; // верхний регистр 1, за длину строки 1, за нижний регистр 1

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ConteinsNumber\_0\_4Points()

{

// Arrange

string password = "Passw0rd";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1

int expected = 4;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ConteinsNumber\_1\_4Points()

{

// Arrange

string password = "Passw1rd";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1

int expected = 4;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

// Tests for special chars

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_at\_5Points()

{

// Arrange

string password = "Passw0rd@";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Hash\_5Points()

{

// Arrange

string password = "Passw0rd#";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Excl\_5Points()

{

// Arrange

string password = "Passw0rd!";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1 int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Doll\_5Points()

{

// Arrange

string password = "Passw0rd$";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

} }

Проверьте работоспособность проектов.

**Задание для самостоятельного выполнения**:

Реализуйте проекты с юнит тестами для проверки всех арифметических операций.