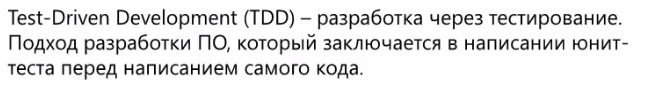
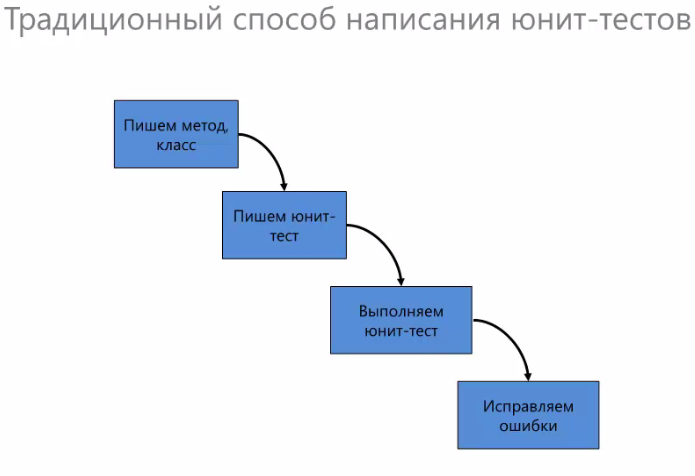
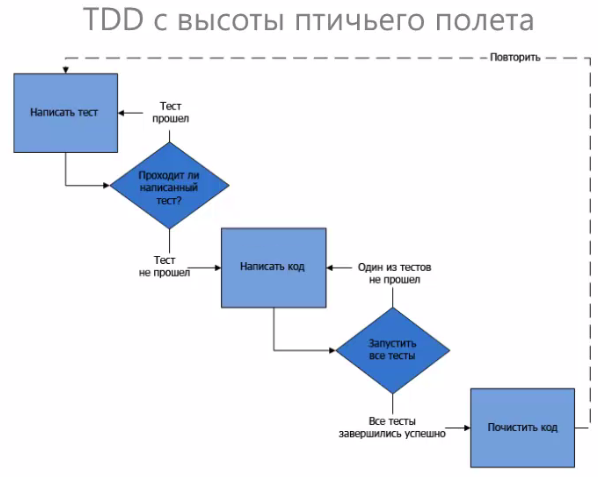
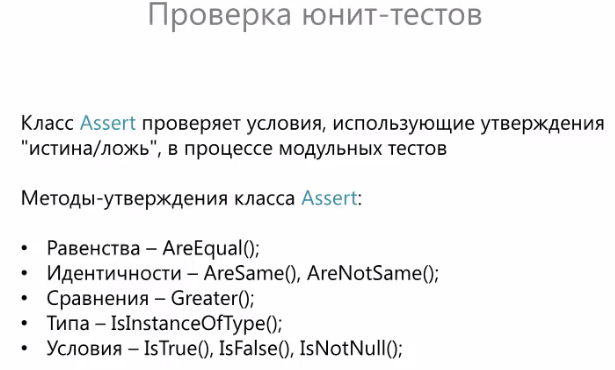
Существует

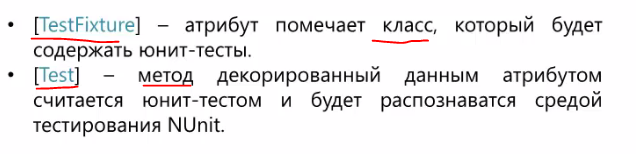
Интеграционное и модульное тестирование

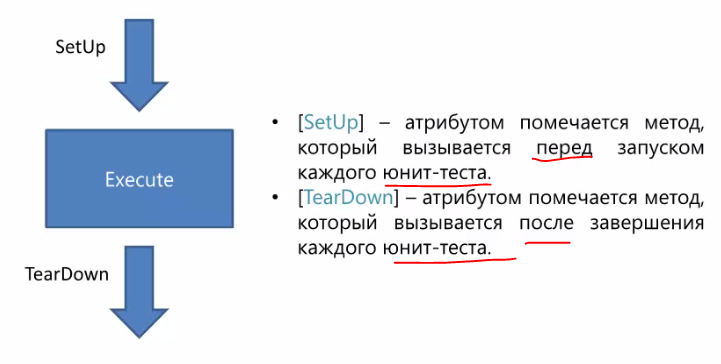


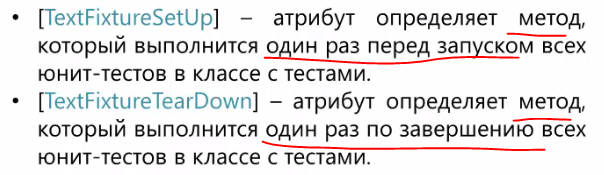




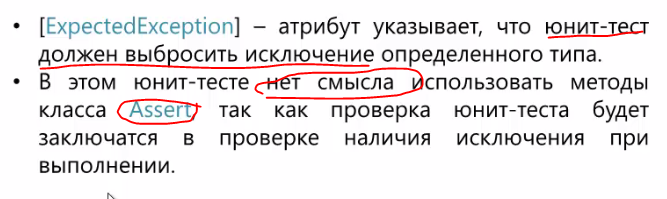




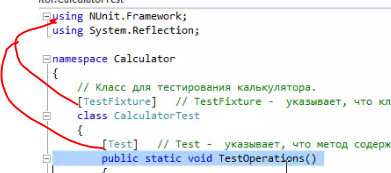




Тест считается пройденным если будет сгенерировано исключение указанного типа в атрибуте



Импортировать пространство имен

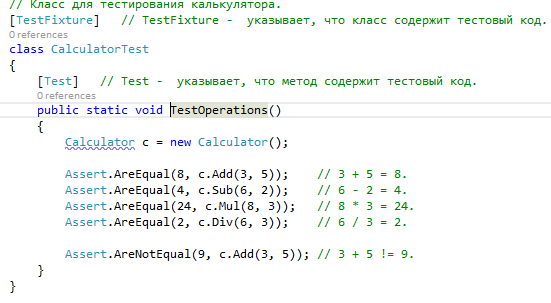


Равно

Assert.AreEqual(8, c.Add(3, 5)); // 3 + 5 = 8.

Не равно

Assert.AreNotEqual(9, c.Add(3, 5)); // 3 + 5 != 9.



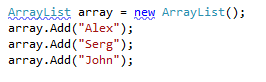
AreSame - проверяет, ссылаются ли переменные на одну и ту же область памяти.

Assert.AreSame(a, b);

Contains используется для проверки объектов,

содержащихся в коллекции или массиве.

Assert.Contains("Alex", array);



Перед прохождением каждого теста будет запущен метод помеченный атрибутом

// Проверяет, является ли один объект больше, чем другой (a > b).

Assert.Greater(a, b);

// a >= b.

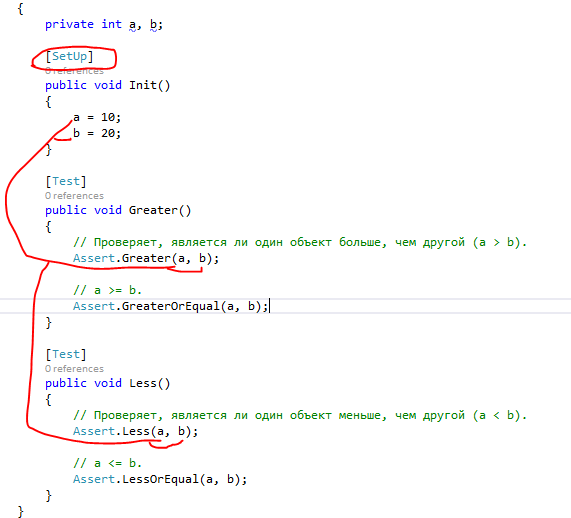
Assert.GreaterOrEqual(a, b);

// Проверяет, является ли один объект меньше, чем другой (a < b).

Assert.Less(a, b);

// a <= b.

Assert.LessOrEqual(a, b);



Методы для проверки типов объектов. (даже если объект наследует данный тип (т.е. можно ли привести к данному типу))

Assert.IsInstanceOf(typeof(object), "Hello"); является

Assert.IsNotInstanceOf(typeof(string), 5); не является

// 2 + 2 = 4 - True.

Assert.IsTrue(calc.Add(2, 2) == 4)

// 3 \* 6 = 20 - False.

Assert.IsFalse(calc.Mul(3, 6) == 20);

double d = double.NaN;

Assert.IsNaN(d); // d is not a number.

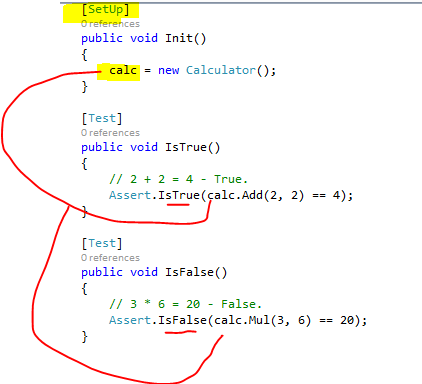
Assert.IsNaN(0 / 0f); // 0/0f is not a number.

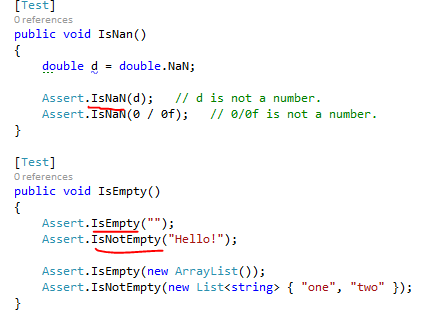
Assert.IsEmpty("");

Assert.IsNotEmpty("Hello!");

Assert.IsEmpty(new ArrayList());

Assert.IsNotEmpty(new List<string> { "one", "two" });





Метод создает тест, который не будет пройден с сообщением Fail!.

Assert.Fail("Fail!");

Этот метод помечает тест, который будет игнорирован при запуске всех тестов

и не будет выполнен. (но только если до указанной строки не будет исключения)

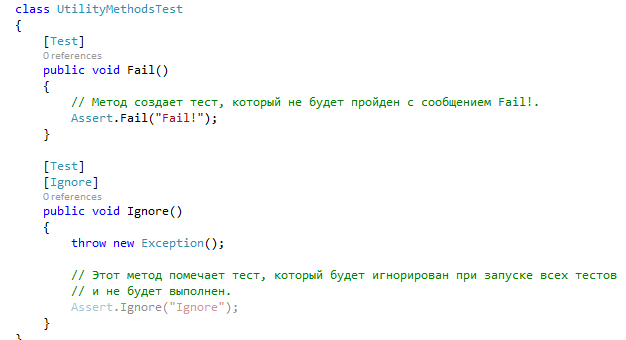
Assert.Ignore("Ignore");

Либо чтобы метод был проигнорирован его следует пометить атрибутом [Ignore]

[Ignore]

public void Ignore()

{…………………….

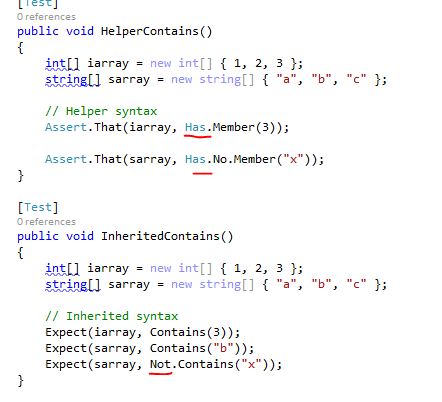


Унаследовав класс AssertionHelper нам предоставляются методы Expect TypeOf Not и т.д.

Assert.That вторым параметром мы можем указать цель проверки



Для элементов коллекции



Метод теста, помеченный атрибутом ExpectedException будет пройден только тогда, когда в его теле будет

обнаружена исключительная ситуация того типа, который указан в параметре этого атрибута

[Test]

[ExpectedException(typeof(DivideByZeroException), ExpectedMessage = "Попытка деления на нуль.")]

public static void TestCalculatorDivideByZeroException()

{

throw new Exception();

new Calculator().Div(1,0);

}

В случае если несколько полей тела метода могут сгенерировать одинаковую исключительную ситуацию то необходимо проверять какое именно поле ее сгенерировало

Можно проверить какое сообщение было отправлено

StringAssert.Contains(ex.Message, "Ctor");

Этот тест проходит только в том случае, если исключительная ситуация ArgumentOutOfRangeException была сгенерированна при

передаче конструктору неправильных параметров

[Test]

public void TestConverterGenareteCtorArgumentException()

{

try

{

UahConverter converter = new UahConverter(0.25, -10, 8);

converter.OutputCurrency = Currency.EUR;

converter.Value = 1000;

}

catch (ArgumentException ex)

{

StringAssert.Contains(ex.Message, "Ctor");

return;

}

Assert.Fail("No exception was thrown.");

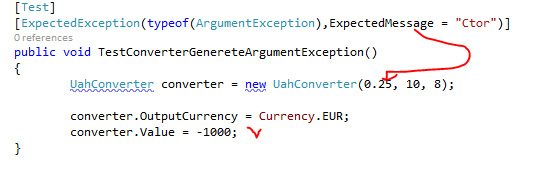
}

Либо

чтобы не проверять try catch то метод помеченный атрибутом и указать какое пользовательское сообщение ждем

[ExpectedException(typeof(ArgumentException),ExpectedMessage = "Ctor")]

Тест не пройдет так как аналогичное исключение будет брошено свойством и у него иное сообщение



Проверка коллекции на наличие хотя б одного элемента

Assert.That(mw.Model.GetModel(),Has.Some.Matches<ProcessInfo>(n=>n.Id==14));