

ОСНОВЫ ТЕСТИРОВАНИЯ

Фоменко Максим Юрьевич





Проблема, о которой ты не думал



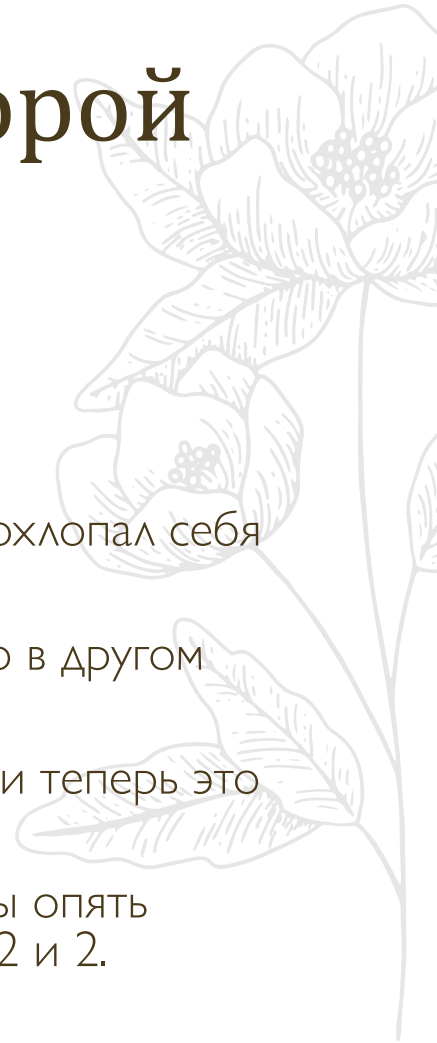
Ты написал код.


Ты его запустил, ввёл $2+2$, получил 4. Похлопал себя по плечу. Молодец.

Потом ты поправил одну строчку где-то в другом месте.

Вопрос: Твои $2+2$ всё ещё работают, или теперь это 22?

Ответ: Ты не знаешь. И чтобы узнать, ты опять запускаешь прогу и вручную вбиваешь 2 и 2.



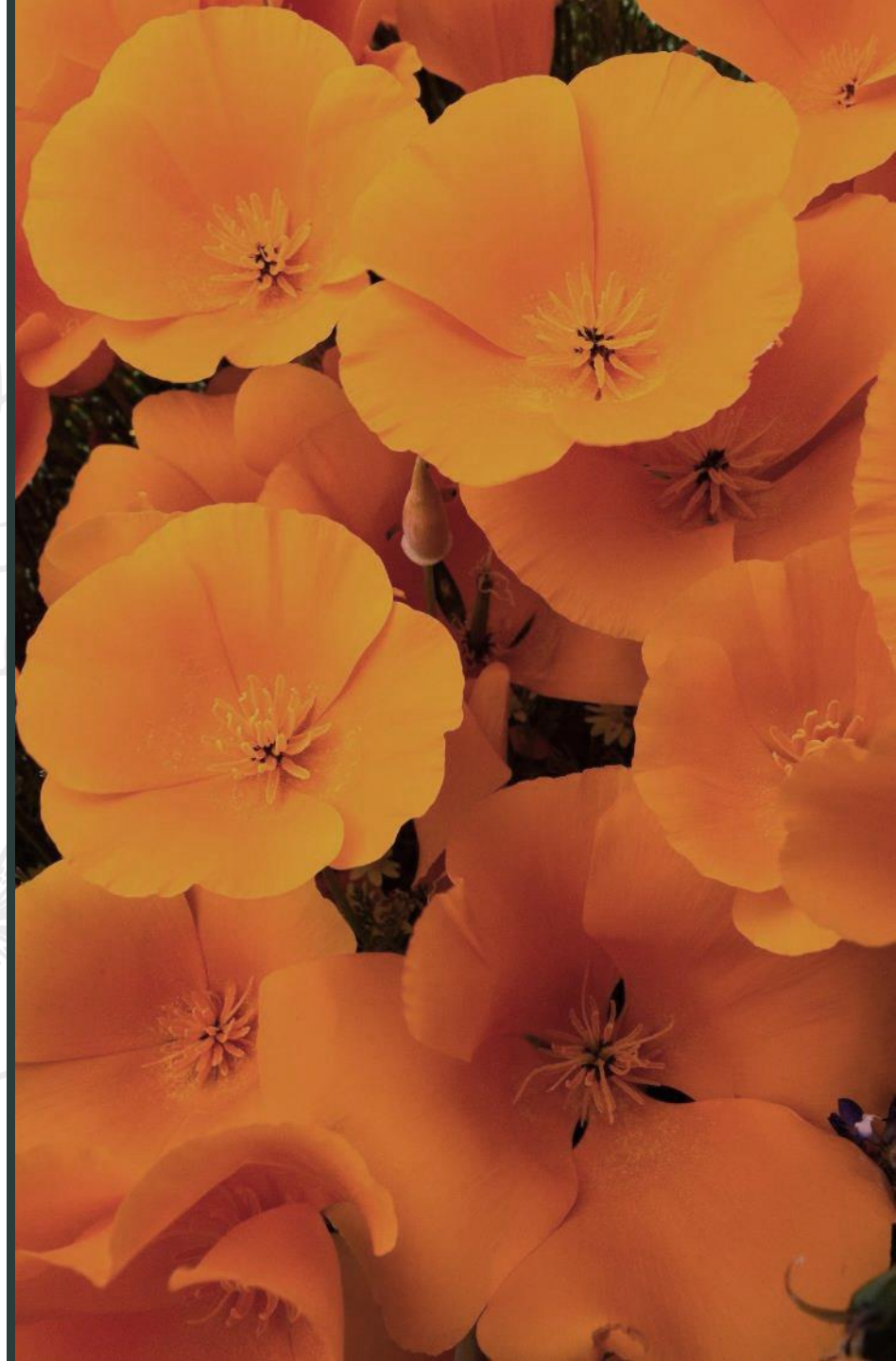


Это называется МАРТЫШКИН
ТРУД. Наша цель — избавиться
от него_{дядя}

Что такое тест?



Тест — это другой кусок кода, который ты пишешь ОДИН РАЗ.



Что такое тест?

Его единственная задача — автоматически проверять твой основной код.

- Он не ленится.
- Он не ошибается.
- Он делает за тебя скучную работу.

Твой личный робот-зануда, который следит, чтобы код работал корректно.



Наш инструмент: MSTest

Забудь все умные слова: NUnit, xUnit, SpecFlow

Кто такой MSTest



Наш (кандидат) инструмент — MSTest

Почему он?

Потому что он **УЖЕ ЕСТЬ** в Visual Studio.

Не надо ничего качать, устанавливать и настраивать.

- Файл
- Создать
- Проект
- Проект модульного теста (.NET).

Всё. Ты в игре.



А А А

Золотое правило

1. Arrange (Подготовка)
2. Act (Действие)
3. Assert (Проверка)

Любой тест
строится по
этой схеме

Sample class

```
Ссылка: 1
class Calculator
{
    Ссылка: 1
    public int Add(int x, int y)
    {
        return x + y;
    }
}
```



Arrange / ПОДГОТОВКА

Задача: подготовить сцену.

- Создай все нужные объекты.
- Задай все нужные переменные.
- Собери все "ингредиенты" для твоего теста.



Arrange / ПОДГОТОВКА

Пример: Если тестируем калькулятор, нам нужны сам калькулятор и два числа.

```
// Arrange  
var calculator = new Calculator();  
int a = 5;  
int b = 10;  
int expectedResult = 15;
```




Act / ДЕЙСТВИЕ

Задача: действуем же ну.

- Вызови тот самый метод, который ты тестируешь.
- Сделай то одно, главное действие.
- Получи результат.



Act / ДЕЙСТВИЕ

Пример: Вызываем метод сложения

```
// Act  
int actualResult = calculator.Add(a, b);
```



Assert / ПРОВЕРКА

Задача: сравнить ожидания с реальностью.

- Берёшь то, что ОЖИДАЛ (expectedResult).
- Берёшь то, что ПОЛУЧИЛ (actualResult).
- Сравниваешь их.



Assert / ПРОВЕРКА

Пример: Проверяем, что результат сложения правильный.

```
// Assert  
Assert.AreEqual(expectedResult, actualResult);
```



testTest


2 мс



А если код должен сломаться?

А зачем?

На слайде знак "опасность", "взрыв"



Иногда правильное поведение кода — это выкинуть ошибку.

Например, `Divide(10, 0)` должно кидать `DivideByZeroException`.

Ссылка: 1

```
class Calculator
```

```
{
```

Ссылка: 1 | 1/1 passing

```
public int Add(int x, int y)
```

```
{
```

```
    return x + y;
```

```
}
```

Ссылка: 0

```
public int Divide(int x, int y)
```

```
{
```

```
    return x / y;
```

```
}
```

```
}
```



```
[TestMethod]
```

```
◆ | Ссылки: 0
```

```
public void Divide_ByZero_ThrowsException()
{
    // Arrange
    var calculator = new Calculator();
    int a = 10;
    int b = 0;

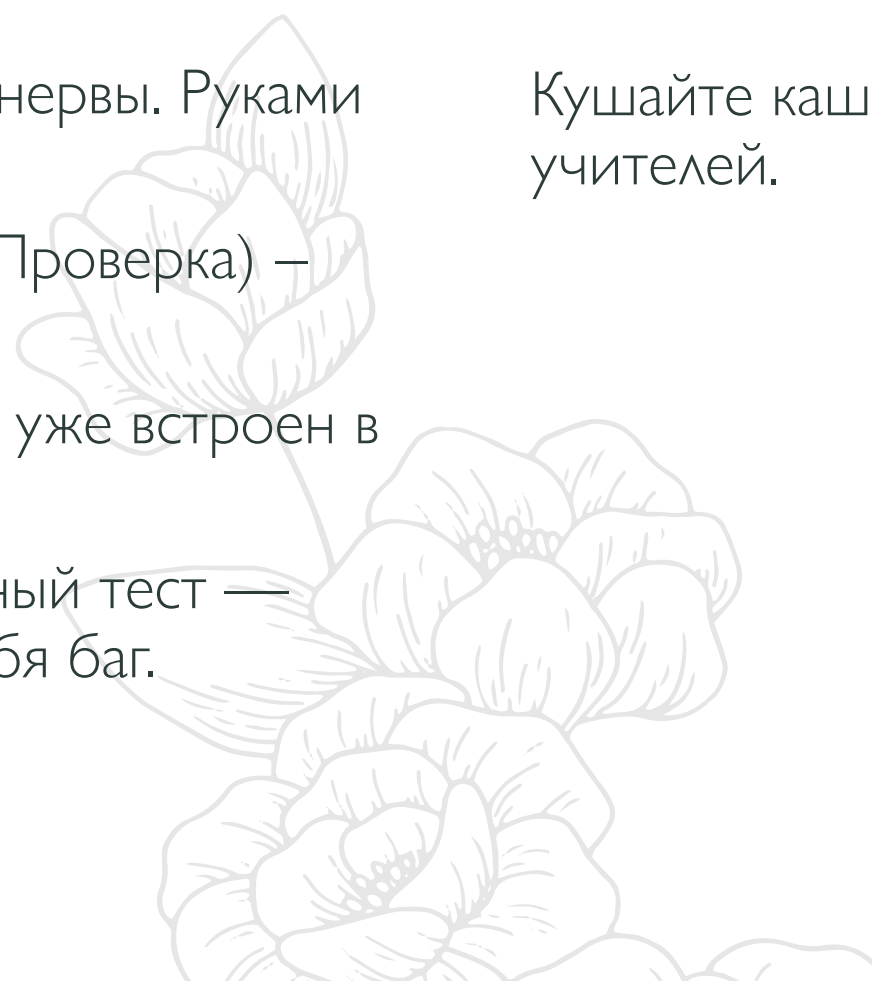
    // Act & Assert
    // Эта конструкция говорит:
    // "Я ожидаю, что код внутри сейчас упадет с ошибкой DivideByZeroException"
    Assert.ThrowsException<DivideByZeroException>(() => calculator.Divide(a, b));
}
```



mmResult

Итоговые советы и рекомендации



- Тесты экономят твоё время и нервы. Руками тестирует только обезьяна.
 - AAA (Подготовка, Действие, Проверка) – база база.
 - MSTest - твой первый друг, он уже встроен в студию.
 - Зелёный тест - хорошо. Красный тест — тоже хорошо, он нашёл за тебя баг.
- 

Кушайте кашу и уважайте учителей.

Спасибо

Fomenko Maksim Yurevich

fo@ma.ru

