ТЕСТИРОВАНИЕ

https://github.com/kontur-courses/**testing**

СКБ Контур

ТЕСТИРОВАНИЕ

- Как сделать тесты максимально полезными

ТЕСТИРОВАНИЕ

- Как сделать тесты максимально полезными
- Как упростить их написание

ТЕСТИРОВАНИЕ

- Как сделать тесты максимально полезными
- Как упростить их написание
- Практика

ТЕСТИРОВАНИЕ

- Кто писал тесты?

ТЕСТИРОВАНИЕ

- Кто писал тесты?
- Зачем нужны тесты?

ТЕСТИРОВАНИЕ

- Кто писал тесты?
- Зачем нужны тесты?
  - Проверять, что код работает правильно

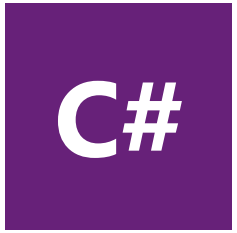
ДОВЕРИЕ ТЕСТАМ

Будет ли тест понятен ревьюеру?

Сможет ли ревьюер быстро убедиться в корректности теста?

ТЕСТЫ КАК СПЕЦИФИКАЦИЯ

```
public class Superman_Should {
        [Test]
        public void SaveKittenFromTree(){
                …
                superman.Act();
                Assert.IsTrue(kitten.IsSaved());
        }
        [Test]
        public void WearRedBlueSuit_WhenAtWork(){
                …
        }
        …
}
```



ТЕСТЫ КАК СПЕЦИФИКАЦИЯ

```
suite("Superman should", () => {
    test("save kitten from tree", () => {
        ...
        superman.act();
        assert.isTrue(kitten.isSaved());
    });

    test("wear red blue suit when at work", () => {
        ...
    });
    ...
});
```

JS

BEHAVIOUR DRIVEN DEVELOPMENT

```
describe("Superman", () => {
    it("should save kitten from tree", () => {
        ...
        superman.act();
        assert.isTrue(kitten.isSaved());
    });

    context("when at work", () => {
        it ("wears red blue suit", () => {
            ...
        });
    });
    ...
});
```

JS

Вопросы?

ТЕСТЫ КАК СПЕЦИФИКАЦИЯ

КАК СДЕЛАТЬ ТЕСТЫ ПОНЯТНЫМИ

- Структура теста
- Именование

ПРАВИЛЬНАЯ СТРУКТУРА ТЕСТА

ПРАВИЛЬНАЯ СТРУКТУРА ТЕСТА

Arrange

ПРАВИЛЬНАЯ СТРУКТУРА ТЕСТА

Arrange

Act

System
Under
Test

ПРАВИЛЬНАЯ СТРУКТУРА ТЕСТА

Arrange

Act

Assert

System
Under
Test

ПРАВИЛЬНАЯ СТРУКТУРА ТЕСТА

```
[Test]
public void GiveResultOfSameSize_OnEqualSizeArrays()
{
    var arr1 = new[] { 1 };
    var arr2 = new[] { 2 };

    var result = arr1.Zip(arr2, Tuple.Create);

    CollectionAssert.AreEqual(
        new[] { Tuple.Create(1, 2) },
        result);
}
```

SAMPLES / AAA / ZIP_SHOULD.CS

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Что должно быть в имени теста?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Что должно быть в имени теста?

1. Conditions: preconditions, input, state

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Что должно быть в имени теста?

1. Conditions: preconditions, input, state
2. System Under Test: class name, method name

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Что должно быть в имени теста?

1. Conditions: preconditions, input, state
2. System Under Test: class name, method name
3. Expected behaviour / Requirement to check

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

ParserTests.TestParse?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

ParserTests.TestParse?

ParserTests.Parse_Fails?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

ParserTests.TestParse?

ParserTests.Parse_Fails?

ParserTests.Parse_BigNumbers?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

ParserTests.TestParse?

ParserTests.Parse_Fails?

ParserTests.Parse_BigNumbers?

ParserTests.Parse_NumbersGreaterThanMaxInt?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

ParserTests.TestParse?

ParserTests.Parse_Fails?

ParserTests.Parse_BigNumbers?

ParserTests.Parse_NumbersGreaterThanMaxInt?

ParserTests.Fail_OnNegativeNumbers?

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Validator_Tests.Validate_WhenNameIsNull_ThrowsBadRequest

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Validator_Tests.Validate_WhenNameIsNull_ThrowsBadRequest

IsAdult_AgeLessThan18_False

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Validator_Tests.Validate_WhenNameIsNull_ThrowsBadRequest

IsAdult_AgeLessThan18_False

ParseInt_Should.Fail_OnNonNumber

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Validator_Tests.Validate_WhenNameIsNull_ThrowsBadRequest

IsAdult_AgeLessThan18_False

ParseInt_Should.Fail_OnNonNumber

Stack_Should.BeEmpty_AfterCreation

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

Validator_Tests.Validate_WhenNameIsNull_ThrowsBadRequest

IsAdult_AgeLessThan18_False

ParseInt_Should.Fail_OnNonNumber

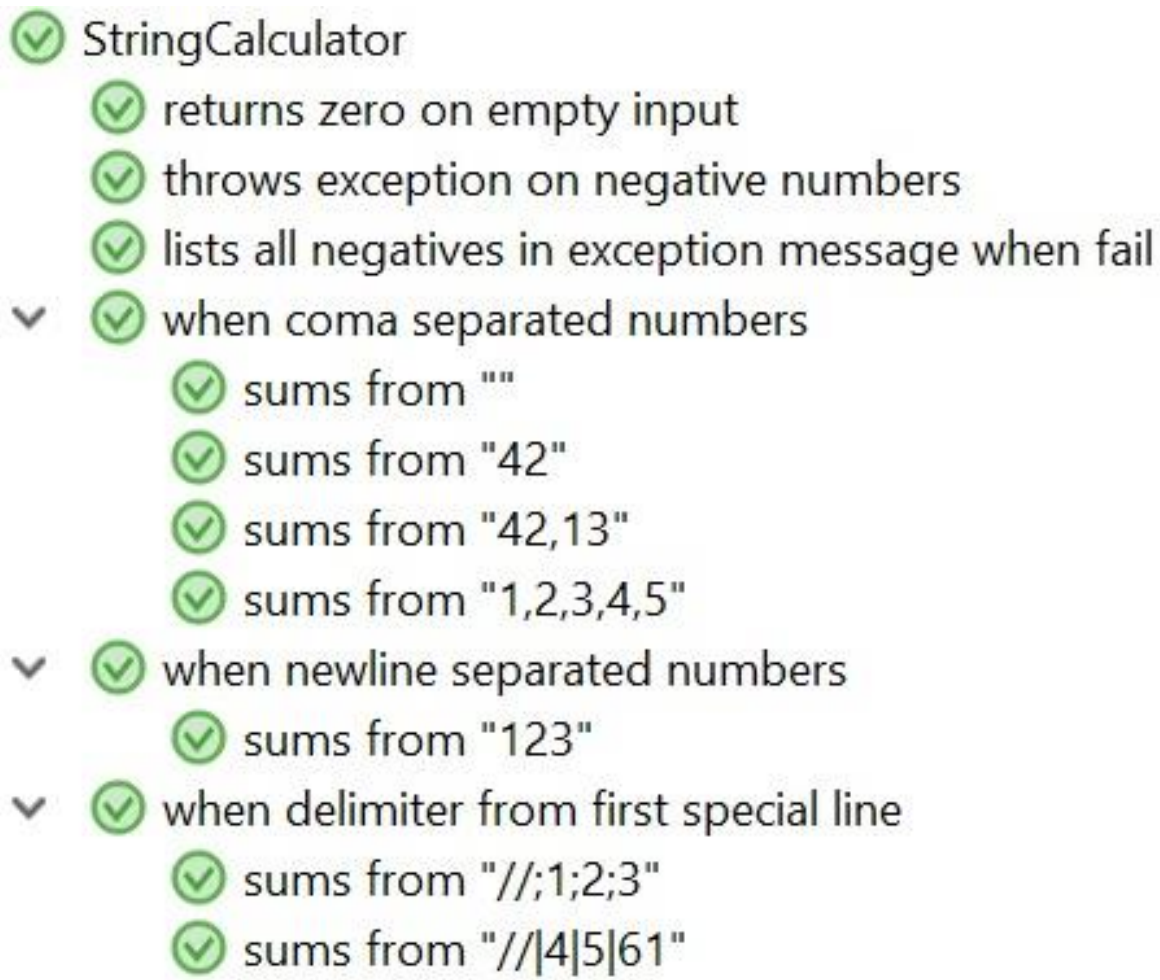
Stack_Should.BeEmpty_AfterCreation

When_MandatoryFieldsAreMissing_Expect_StudentAdmissionToFail

ИМЯ ТЕСТА КАК СПЕЦИФИКАЦИЯ

```
public class Stack_Specification
{
    public void Constructor_CreatesEmptyStack() {...}
    public void Constructor_PushesItemsToEmptyStack()
    public void ToArray_ReturnsItemsInPopOrder()
    public void Push_AddsItemToStackTop() {...}
    public void Pop_OnEmptyStack_Fails() {...}
    public void Pop_ReturnsLastPushedItem() {...}
}
```

SAMPLES / SPECIFICATIONS / STACK_SPECIFICATION.CS



ПРИМЕР СПЕЦИФИКАЦИИ ТЕСТАМИ

Вопросы?

СТРУКТУРА И ИМЕНОВАНИЕ

АНТИПАТТЕРНЫ

SAMPLES / ANTIPATTERNS

АНТИПАТТЕРНЫ

SAMPLES / ANTIPATTERNS

Local Hero

Loudmouth

Free Ride

Over specification

Вопросы?

АНТИПАТТЕРНЫ

ТЕСТ НАПИСАТЬ – КАК ЧАЙ ПОПИТЬ

ПИШЕМ ТЕСТЫ ЛЕГКО

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения
- Создание тестовых данных
- Похожие тесты
- Похожие проверки
- Разметка тестов
- Запуск тестов

СБОРКА И РАЗБОРКА ОКРУЖЕНИЯ

C#

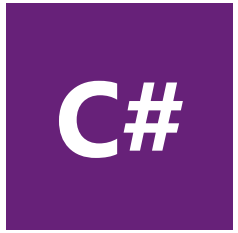
OneTimeSetUp
SetUp
Test 1
TearDown
SetUp
Test 2
TearDown
OneTimeTearDown

JS

before
beforeEach
test 1
afterEach
beforeEach
test 2
afterEach
after

SETUP & TEARDOWN

```
[TestFixture]
public class Mailbox_Should
{
    private Mailbox mailbox;
    [SetUp]
    public void SetUp() { mailbox = new Mailbox(); }
    ...
    [TearDown]
    public void TearDown() { mailbox.Dispose(); }
}
```



BEFORE & AFTER

```
describe("Mailbox", () => {
    let mailbox;
    beforeEach(() => {
        mailbox = new Mailbox();
    });
    ...
    afterEach(() => {
        mailbox.dispose();
    });
});
```

JS

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных
- Похожие тесты
- Похожие проверки
- Разметка тестов
- Запуск тестов

OBJECT MOTHER & TEST DATA BUILDER

SAMPLES / TESTDATABUILDER / TESTDATABUILDER.CS

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных → TestUsers.cs
- Похожие тесты
- Похожие проверки
- Разметка тестов
- Запуск тестов

PARAMETRIZED TESTS

Они же Data Driven

C# Через атрибуты TestCase и TestCaseSource

SAMPLES / PARAMETRIZED / DOUBLE_SHOULD.CS

JS Динамически генерируемые тесты

SAMPLES / PARAMETRIZED / NUMBER_SHOULD.JS

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных → TestUsers.cs
- Похожие тесты → Parametrized Tests
- Похожие проверки
- Разметка тестов
- Запуск тестов

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных → TestUsers.cs
- Похожие тесты → Parametrized Tests
- Похожие проверки → Свои Assert-ы
- Разметка тестов
- Запуск тестов

LIVE TEMPLATES

C#

Открыть Resharper → Tools → Templates Explorer
Импортировать tests-templates.DotSettings

tf — TestFixture
tt — Test
su — SetUp

JS

Копировать Mocha.xml в %USERPROFILE%\.WebStormNN\config\templates
Открыть File → Settings → Editor → Live Templates

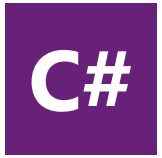
desc — describe
it — it
before — beforeEach

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных → TestUsers.cs
- Похожие тесты → Parametrized Tests
- Похожие проверки → Свои Assert-ы
- Разметка тестов → Live Templates
- Запуск тестов

HOTKEYS



Ctrl+T+R или Ctrl+U+R — Run tests

Alt+Shift+R — Rerun tests

ЧТО ПРЕВРАЩАЕТ ТЕСТЫ В РУТИНУ?

Повторяющиеся действия:

- Подготовка окружения → [SetUp], beforeEach
- Создание тестовых данных → TestUsers.cs
- Похожие тесты → Parametrized Tests
- Похожие проверки → Свои Assert-ы
- Разметка тестов → Live Templates
- Запуск тестов → Hotkeys

Вопросы?

БОРЬБА С ДУБЛИРОВАНИЕМ

ДОПОЛНИТЕЛЬНЫЕ ТРЮКИ

- Should вместо Assert
- Ожидание исключения
- Ограничение по времени
- Выбор тестов для прогона

SHOULD BMECTO ASSERT

SHOULD ВМЕСТО ASSERT

1. `Assert.AreEqual(expected, actual)` или `Assert.AreEqual(actual, expected)`?

SHOULD ВМЕСТО ASSERT

1. `Assert.AreEqual(expected, actual)` или `Assert.AreEqual(actual, expected)`?
2. `Assert` — корявая семантика
   `Assert.That(2+2, Is.EqualTo(4))`

SHOULD ВМЕСТО ASSERT

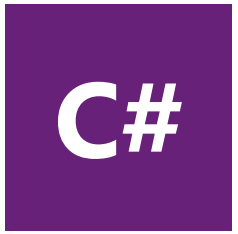
1. `Assert.AreEqual(expected, actual)` или `Assert.AreEqual(actual, expected)`?
2. `Assert` — корявая семантика
   `Assert.That(2+2, Is.EqualTo(4))`
3. Неудачное API:
   `Assert.That(x, IResolveConstraint ?!?) // O_o`

SHOULD

1. `(2+2).Should().Be(4)`
2. `flag.Should().BeTrue()`
3. Для массивов
    1. ```
       new[] {1,2,3}
           .Should()
           .BeEquivalentTo(new[] {3,2,1});
       ```
       без учёта порядка!

    2. ```
       new[] {1,2,3}
           .Should().Equal(new[] {1,2,3});
       ```

FluentAssertions – доступна через NuGet



SHOULD И EXPECT

```
expect(2+2).to.equal(4);
(2+2).should.be.equal(4);

let flag = true;
expect(flag).to.be.true;
flag.should.be.true;

expect([1, 2, 3]).to.be.eql([1, 2, 3]);
[1, 2, 3].should.be.eql([1, 2, 3]);
```

to и be – можно безболезненно убирать

Chai поддерживает стиль should и expect

JS

ОЖИДАНИЕ ИСКЛЮЧЕНИЯ

C#

```
Action action = () => { var z = x / y; };
action.ShouldThrow<DivideByZeroException>();

Assert.Throws<DivideByZeroException>(
 () => { var z = x / y });
```

JS

```
expect(() => {
    z = x / y
}).to.throw();
```

ОГРАНИЧЕНИЕ ПО ВРЕМЕНИ

C#

```
[Test, Timeout(1000)]
public void ShouldDoInTimeout()
{
    …
}
```

JS

```
it("should do in timeout", () => {
    …
}).timeout(1000);
```

ВЫБОР ТЕСТОВ ДЛЯ ПРОГОНА

JS

```
it.only("only you", () => {...});

it.skip("crashed test", () => {...});
```

Вопросы?

ДОПОЛНИТЕЛЬНЫЕ ТРЮКИ

CHALLENGE

CHALLENGE

В проекте Challenge в файле WordsStatistics_Tests напишите тесты:

1. WordsStatistics — должен проходить все тесты.
2. WordStatisticsXXX —скрытые некорректные реализации. Должны падать хотя бы на одном тесте.

Запускайте по Ctrl+F5.

CHALLENGE

Открываем DoNotOpen!

РАЗБОР CHALLENGE

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

Про взаимодействие разных пунктов спецификации подумать трудно (E3)

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

Про взаимодействие разных пунктов спецификации подумать трудно (E3)

Про тесты на производительность вспомнить труднее (998, 999)

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

Про взаимодействие разных пунктов спецификации подумать трудно (E3)

Про тесты на производительность вспомнить труднее (998, 999)

Тесты не заменяют Code Review (STA)

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

Про взаимодействие разных пунктов спецификации подумать трудно (E3)

Про тесты на производительность вспомнить труднее (998, 999)

Тесты не заменяют Code Review (STA)

Code Review не заменяет тесты (CR)

РАЗБОР CHALLENGE

Тесты по спецификации — это просто

Про взаимодействие разных пунктов спецификации подумать трудно (E3)

Про тесты на производительность вспомнить труднее (998, 999)

Тесты не заменяют Code Review (STA)

Code Review не заменяет тесты (CR)

Большие цифры в лидерборде — плохо (Overspecification)

ОБРАТНАЯ СВЯЗЬ

Заполни форму обратной связи по ссылке

http://bit.ly/kontur-courses-feedback

или

по ярлыку *feedback* в корне репозитория