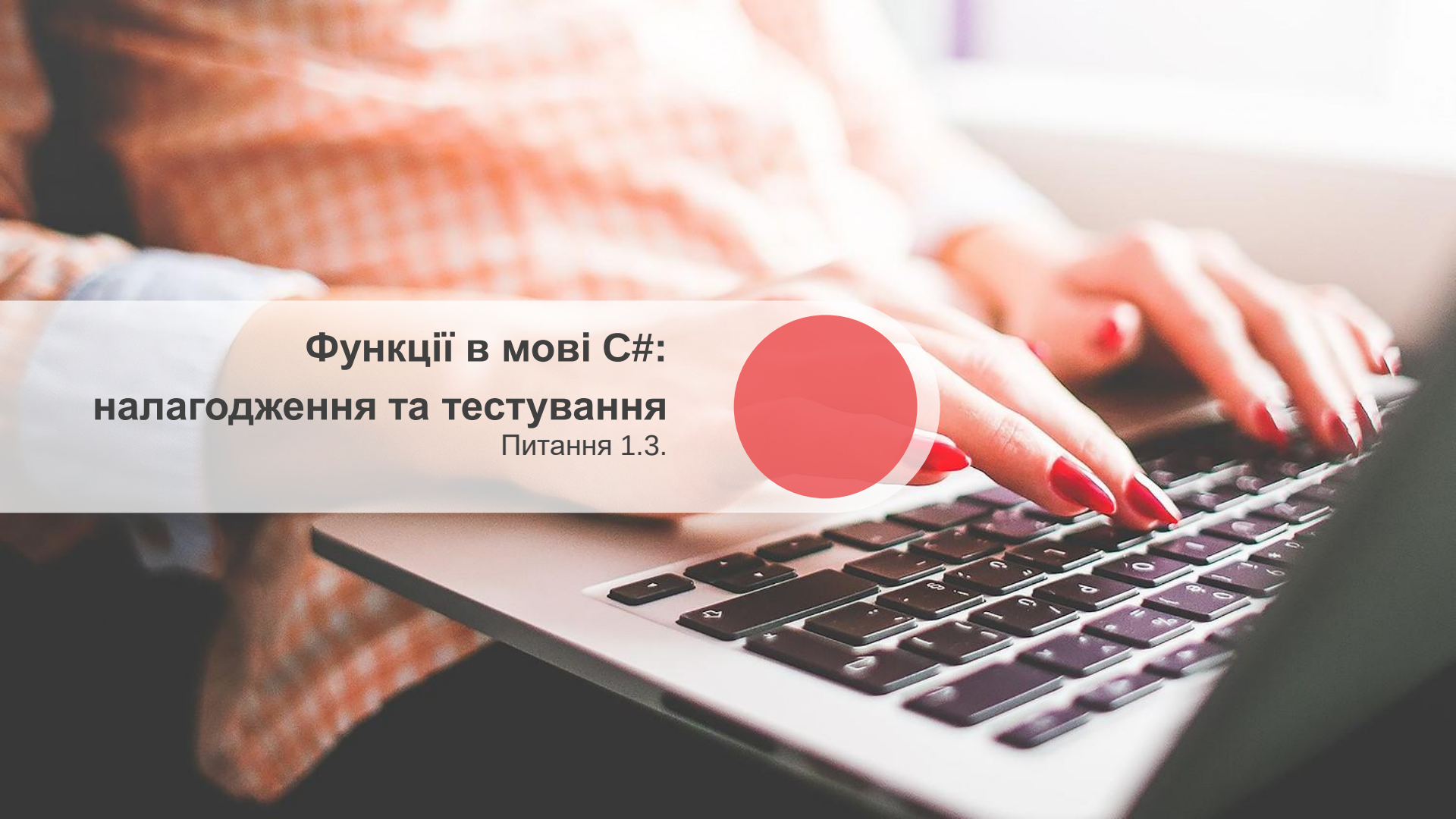


Функції в мові C#: налагодження та тестування

Питання 1.3.

Можливості налагоджувальника Visual Studio 2019

- Покрокове виконання коду
- Точки зупинки (breakpoints)
- Журналювання ходу виконання
- Прослуховувачі Debug та Trace

https://michaelscodingspot.com/debugging-part1/
https://michaelscodingspot.com/debugging-part2/
https://michaelscodingspot.com/debugging-exceptions/

https://docs.microsoft.com/en-us/visualstudio/debugger/debugging-absolute-beginners?view=vs-2019
https://docs.microsoft.com/en-us/visualstudio/debugger/write-better-code-with-visual-studio?view=vs-2019

Автоматизоване тестування

та його переваги

- Test your code frequently, in less time
- Catch bugs before deploying
- Deploy with confidence
- Refactor with confidence
- Focus more on the quality

Типи тестів

Модульні, інтеграційні, наскрізні (end-to-end)

UNIT TEST

Tests a unit of an application without its **external dependencies**

Cheap to write

Execute fast

Don't give a lot of confidence

INTEGRATION TEST

Tests the application with its **external dependencies**

Take longer to execute

Give more confidence

END-TO-END TEST

Drives an application through its UI.

Give you the greatest confidence

Very slow

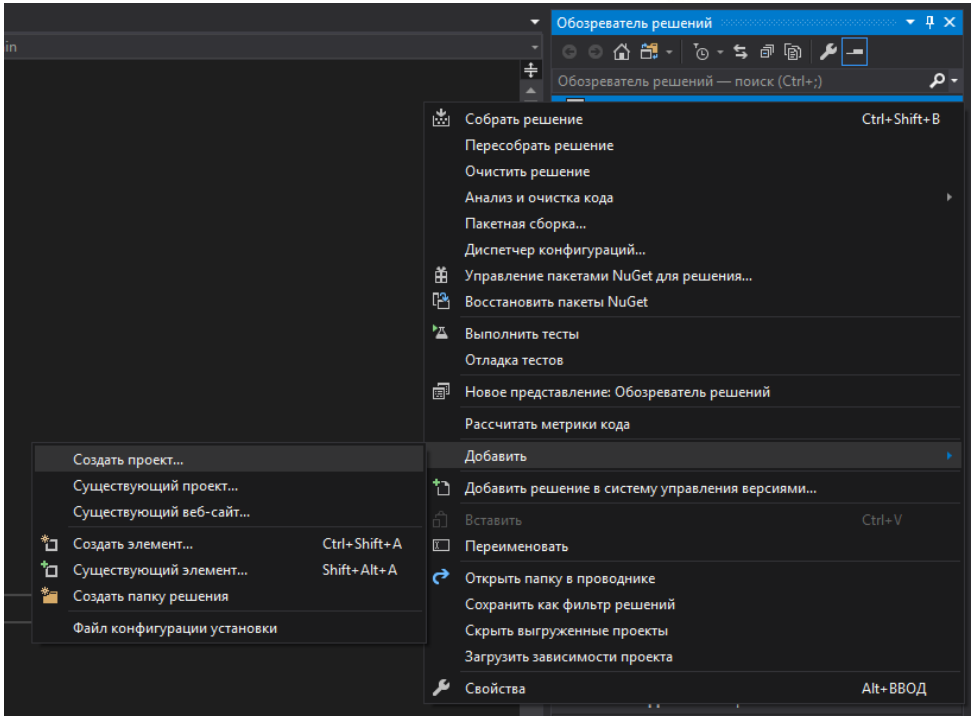
Very brittle

Інструменти тестування для C#-коду

NUnit, MSTest, xUnit, ReSharper, Rider

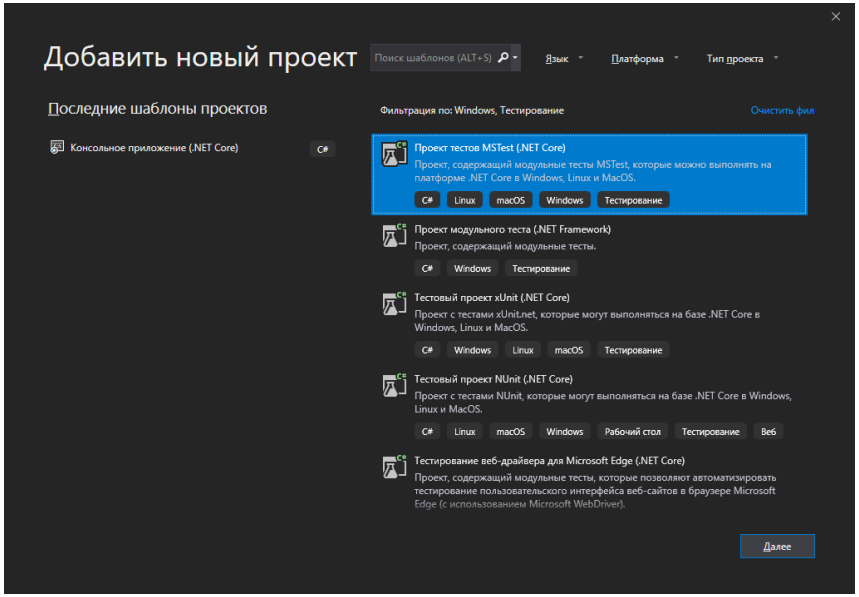
Нехай пишеться додаток для бронювання столика в ресторані

```
1  namespace TestNinja.Fundamentals
2  {
3      public class Reservation
4      {
5          public User MadeBy { get; set; }
6
7          public bool CanBeCancelledBy(User user)
8          {
9              return (user.IsAdmin || MadeBy == user);
10         }
11
12     }
13
14     public class User
15     {
16         public bool IsAdmin { get; set; }
17     }
18 }
```



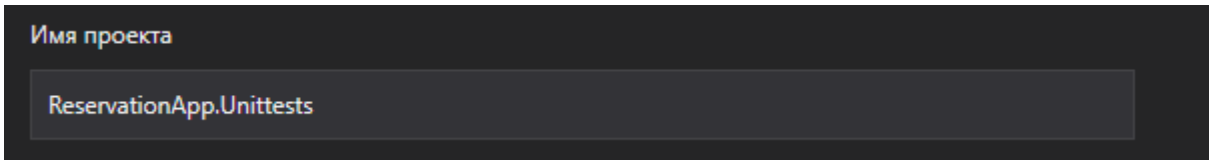
Пишемо тестовий проект

Додамо новий MSTest-проект




```
1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4  namespace TestNinja.UnitTests
5  {
6      [TestClass]
7      public class ReservationTests
8      {
9          [TestMethod]
10         public void CanBeCancelledBy_UserIsAdmin_ReturnsTrue()
11         {
12         }
13     }
14 }
15
```

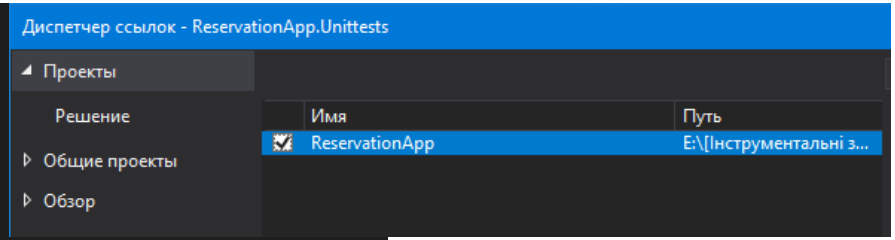
Іменування тестового випадку:
НазваМодуля_Сценарій_Очікуваний результат

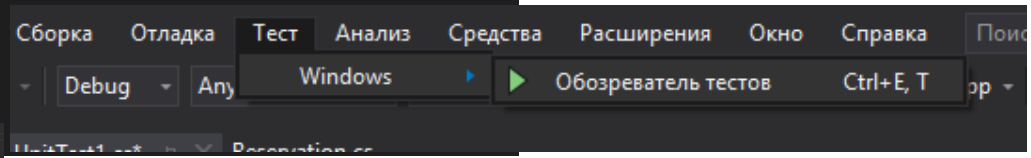


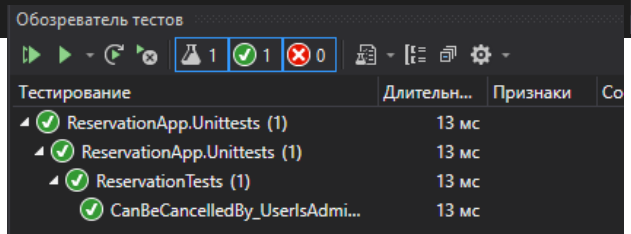
Написання тестового випадку

Принцип Arrange – Act – Assert

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace ReservationApp.Unittests
{
    [TestClass]
    public class ReservationTests
    {
        [TestMethod]
        public void CanBeCancelledBy_UserIsAdmin_ReturnsTrue()
        {
            // Arrange
            var reservation = new Reservation();

            // Act
            var result = reservation.CanBeCancelledBy(new User { IsAdmin = true });

            // Assert
            Assert.IsTrue(result);
        }
    }
}
```



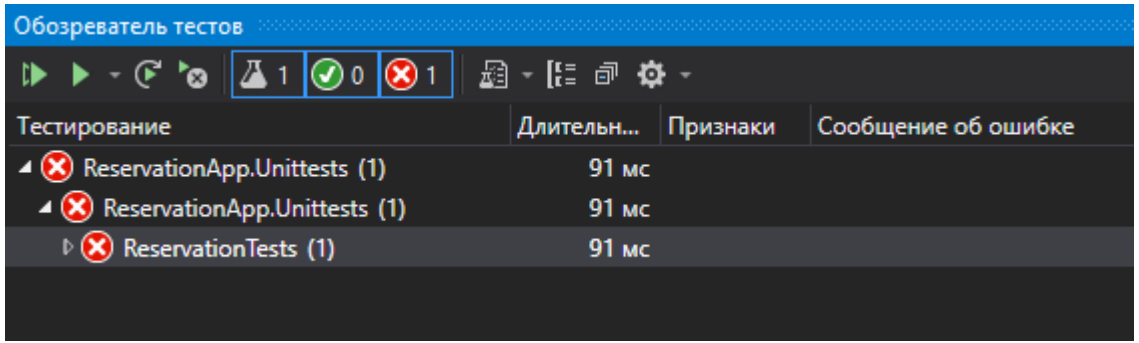




Написання тестового випадку

Імітуємо помилку

```
public bool CanBeCancelledBy(User user)
{
    return !(user.IsAdmin || MadeBy == user);
}
```



Практика: дописати решту тестових випадків

Доповіді: Тестування за допомогою фреймворку NUnit (включити цей приклад)

Тестування за допомогою фреймворку xUnit (включити цей приклад)

Характеристики хорошого модульного тесту

Простота, відсутність упраляючих інструкцій, ізольованість від інших тестів

- Тестовані функції/методи можна поділити на 2 види:
  - Query-функції: повертають деяке значення; тест перевіряє, чи відповідає це значення очікуваному результату; різні ходи виконання (execution path) – різні тестові сценарії;
  - Command-функції: виконують зміну в системі – дію (action – зміна стану об'єкта, запис у БД, виклик веб-служби, відправка повідомлення в чергу повідомлень тощо); можуть також повертати значення;
    - При зміні стану перевіряють поточний стан об'єкта
    - При підключенні зовнішніх ресурсів перевіряють правильність виклику (звернення) за посиланням на ці ресурси (external dependencies)
- Ніколи не тестуються:
  - Language features
  - Сторонній код (3rd-party code)
- Тести підганяються під вимоги, а не під реалізований код!

Атрибути фреймворків для тестування

Nunit, MSTest 2.x, xUnut 2.x

NUnit	MSTest 2.x	xUnit 2.x	Коментарі
[Test]	[TestMethod]	[Fact]	Позначає тестовий метод
[TestFixture]	[TestClass]	--	Позначає тестовий клас
[SetUp]	[TestInitialize]	Конструктор	Спрацьовує перед кожним тес товим випадком
[TearDown]	[TestCleanup]	IDisposable. Dispose	Спрацьовує після кожного тес тового випадку
[OneTimeSetUp]	[ClassInitialize]	IClassFixture<T>	Метод спрацьовує 1 раз перед запуском тестового набору
[OneTimeTearDown]	[ClassCleanup]	IClassFixture	Метод спрацьовує 1 раз після запуску тестового набору

Атрибути фреймворків для тестування

Nunit, MSTest 2.x, xUnut 2.x

NUnit	MSTest 2.x	xUnit 2.x	Коментарі
[Ignore("reason")]	[Ignore]	[Fact(Skip="reason")]	Пропуск тестового випадку
[Property]	[TestProperty]	[Trait]	Встановлює довільні мета дані для тесту
[Theory]	[DataRow]	[Theory]	Конфігурує керовані дани ми (data-driven) тести
[Category("")]	[TestCategory("")]	[Trait("Category", "")]	Категоризує тестові випад ки або класи

Test Execution Workflow

Атрибути для налаштування тестів

```
[TestClass]
public class YourUnitTests
{
    [AssemblyInitialize]
    public static void AssemblyInit(TestContext context)
    {
        // Executes once before the test run. (Optional)
    }
    [ClassInitialize]
    public static void TestFixtureSetup(TestContext context)
    {
       // Executes once for the test class. (Optional)
    }

    [TestInitialize]
    public void Setup()
    {
        // Runs before each test. (Optional)
    }
    [AssemblyCleanup]
    public static void AssemblyCleanup()
    {
       // Executes once after the test run. (Optional)
    }
    [ClassCleanup]
    public static void TestFixtureTearDown()
    {
        // Runs once after all tests in this class are executed.
(Optional)
        // Not guaranteed that it executes instantly after all tests
from the class.
    }

    [TestCleanup]
    public void TearDown()
    {
        // Runs after each test. (Optional)
    }
    // Mark that this is a unit test method. (Required)
    [TestMethod]
    public void YouTestMethod()
    {
       // Your test code goes here.
    }
}
```

Класс может включать любое количество членов (таких как конструкторы, свойства, методы и события) и элементов данных (полей).

Assertions

```
Assert.AreEqual(28, _actualFuel); // Tests whether the specified values are equal.
Assert.AreNotEqual(28, _actualFuel); // Tests whether the specified values are unequal. Same as AreEqual for numeric values.
Assert.AreSame(_expectedRocket, _actualRocket); // Tests whether the specified objects both refer to the same object
Assert.AreNotSame(_expectedRocket, _actualRocket); // Tests whether the specified objects refer to different objects
Assert.IsTrue(_isThereEnoughFuel); // Tests whether the specified condition is true
Assert.IsFalse(_isThereEnoughFuel); // Tests whether the specified condition is false
Assert.IsNull(_actualRocket); // Tests whether the specified object is null
Assert.IsNotNull(_actualRocket); // Tests whether the specified object is non-null
Assert.IsInstanceOfType(_actualRocket, typeof(Falcon9Rocket)); // Tests whether the specified object is an instance of the expected type
Assert.IsNotInstanceOfType(_actualRocket, typeof(Falcon9Rocket)); // Tests whether the specified object is not an instance of type
StringAssert.Contains(_expectedBellatrixTitle, "Bellatrix"); // Tests whether the specified string contains the specified substring
StringAssert.StartsWith(_expectedBellatrixTitle, "Bellatrix"); // Tests whether the specified string begins with the specified substring
StringAssert.Matches("(281)388-0388", @"(?d{3})?-? *d{3}-? *-?d{4}"); // Tests whether the specified string matches a regular expression
StringAssert.DoesNotMatch("281)388-0388", @"(?d{3})?-? *d{3}-? *-?d{4}"); // Tests whether the specified string does not match a regular expression
CollectionAssert.AreEqual(_expectedRockets, _actualRockets); // Tests whether the specified collections have the same elements in the same order and quantity.
CollectionAssert.AreNotEqual(_expectedRockets, _actualRockets); // Tests whether the specified collections does not have the same elements or the elements are in a different order and quantity.
CollectionAssert.AreEquivalent(_expectedRockets, _actualRockets); // Tests whether two collections contain the same elements.
CollectionAssert.AreNotEquivalent(_expectedRockets, _actualRockets); // Tests whether two collections contain different elements.
CollectionAssert.AllItemsAreInstancesOfType(_expectedRockets, _actualRockets); // Tests whether all elements in the specified collection are instances of the expected type
CollectionAssert.AllItemsAreNotNull(_expectedRockets); // Tests whether all items in the specified collection are non-null
CollectionAssert.AllItemsAreUnique(_expectedRockets); // Tests whether all items in the specified collection are unique
CollectionAssert.Contains(_actualRockets, falcon9); // Tests whether the specified collection contains the specified element
CollectionAssert.DoesNotContain(_actualRockets, falcon9); // Tests whether the specified collection does not contain the specified element
CollectionAssert.IsSubsetOf(_expectedRockets, _actualRockets); // Tests whether one collection is a subset of another collection
CollectionAssert.IsNotSubsetOf(_expectedRockets, _actualRockets); // Tests whether one collection is not a subset of another collection
Assert.ThrowsException<ArgumentNullException>(() => new Regex(null)); // Tests whether the code specified by delegate throws exact given exception of type T
```

Конкретні та загальні тести

Тестування рядків

- Тестований код

```
1  namespace TestNinja.Fundamentals
2  {
3      public class HtmlFormatter
4      {
5          public string FormatAsBold(string content)
6          {
7              return $"<strong>{content}</strong>";
8          }
9      }
10 }
```

- Конкретний тест:
  - Assert.That(result, Is.EqualTo("abc"));
- Загальний тест:
  - Assert.That(result, Does.StartWith(""));
    Assert.That(result, Does.EndWith(""));
    Assert.That(result, Does.Contain("abc"));

Тестування рядків

Повний тест в NUnit

```
[TestFixture]
public class HtmlFormatterTests
{
    [Test]
    public void FormatAsBold_WhenCalled_ShouldEncloseStringWithStrongElement()
    {
        var formatter = new HtmlFormatter();

        var result = formatter.FormatAsBold("abc");

        // this is specific: in this case, specific is good
        // in some cases, it may be more appropriate to be more general

        Assert.That(result, Is.EqualTo("<strong>abc</strong>"));

        // you can also ignore case
        Assert.That(result, Is.EqualTo("<strong>ABC</strong>").IgnoreCase);

        // examples of more general assertions that aren't good enough in this case
        // but they give you some ideas of how to be more general
        Assert.That(result, Does.StartWith("<strong>"));
        Assert.That(result, Does.EndWith("</strong>"));
        Assert.That(result, Does.Contain("abc"));
    }
}
```

Конкретні та загальні тести

Тестування масивів та колекцій

- Тестований код

```
public class Math
{
    public int Add(int a, int b)
    {
        return a + b;
    }

    public int Max(int a, int b)
    {
        return (a > b) ? a : b;
    }

    public IEnumerable<int> GetOddNumbers(int limit)
    {
        for (var i = 0; i <= limit; i++)
            if (i % 2 != 0)
                yield return i;
    }
}
```

- Налаштування тестів

```
private Math math;

[SetUp]
protected void SetUp()
{
    math = new Math();
}
```

- var result = _math.GetOddNumbers(5);
- Загальні тести:
  - Assert.That(result, Is.Not.Empty);
  - Assert.That(result.Count(), Is.EqualTo(3);
- Конкретний тест:
  - Assert.That(result, Is.EquivalentTo(new [] {1, 3, 5}));

Тестування масивів та колекцій

Загальний вигляд можливого тесту (NUnit)

```
[Test]
public void GetOddNumbers_LimitIsGreaterThanZero_ReturnOddNumbersUpToLimit()
{
    var result = _math.GetOddNumbers(5);

      Assert.That(result, Is.Not.Empty);

      Assert.That(result.Count(), Is.EqualTo(3));

      Assert.That(result, Does.Contain(1));
      Assert.That(result, Does.Contain(3));
      Assert.That(result, Does.Contain(5));

    Assert.That(result, Is.EquivalentTo(new [] {1, 3, 5}));

      Assert.That(result, Is.Ordered);
      Assert.That(result, Is.Unique);

}
```

Конкретні та загальні тести

Перевірка типу значення, яке повертає метод

- Тестований код

```
public class CustomerController
{
    public ActionResult GetCustomer(int id)
    {
        if (id == 0)
            return new NotFound();

        return new Ok();
    }
}

public class ActionResult { }

public class NotFound : ActionResult { }

public class Ok : ActionResult { }
```

```
private CustomerController _controller;

[SetUp]
public void SetUp()
{
    _controller = new CustomerController();
}

[Test]
public void GetCustomer_IdIsZero_ReturnNotFound()
{
    var result = _controller.GetCustomer(0);

    // Not Found
    Assert.That(result, Is.TypeOf<NotFound>());

    // NotFound or one of its derivatives
    //Assert.That(result, Is.InstanceOf<NotFound>());
}

[Test]
public void GetCustomer_IdIsNotZero_ReturnOk()
{
    var result = _controller.GetCustomer(1);

    Assert.That(result, Is.TypeOf<Ok>());
}
```

- Код має 2 шляхи виконання

Конкретні та загальні тести

Тестування void-методів

- Зазвичай такими є command-функції
- Тестований код:

```
public string LastError { get; set; }

public event EventHandler<Guid> ErrorLogged;

public void Log(string error)
{
    if (String.IsNullOrWhiteSpace(error))
        throw new ArgumentNullException();

    LastError = error;

    // Write the log to a storage
    // ...

    ErrorLogged?.Invoke(this, Guid.NewGuid());
}
```

- Тестується зміна стану об'єкта (властивості LastError)

```
private ErrorLogger _logger;

[SetUp]
public void SetUp()
{
    _logger = new ErrorLogger();
}

[Test]
public void Log_WhenCalled_SetTheLastErrorProperty()
{
    _logger.Log("a");

    Assert.That(_logger.LastError, Is.EqualTo("a"));
}
```

Конкретні та загальні тести

Тестування методів, які викидають виключення

- Тестуються варіанти: null, “”, “ “. Маємо параметризований тест
- Для тестування методів, які викидають виключення, використовують делегати або лямбда-вирази

```
[Test]
[TestCase(null)]
[TestCase("")]
[TestCase(" ")]
public void Log_InvalidError_ThrowArgumentNullException(string error)
{
    Assert.That(() => _logger.Log(error), Throws.ArgumentNullException);
}
```

Конкретні та загальні тести

Тестування методів, які викликають (raise) подію

- Перевіримо спрацювання події ErrorLogged
  - Спочатку підписуємось на подію
  - Перевіряємо зміну значення id (Not Empty) після обробки події

```
[Test]
public void Log_ValidError_RaiseErrorLoggedEvent()
{
    var id = Guid.Empty;
    _logger.ErrorLogged += (sender, args) => { id = args; };

    _logger.Log("a");

    Assert.That(id, Is.Not.EqualTo(Guid.Empty));
}
```

Конкретні та загальні тести

Тестування приватних та захищених методів

- Приватні та захищені атрибути класу описують деталі реалізації.
  - Якщо написати тест до них, вони з ним стануть зв'язаними (coupling)
  - Зміна деталей реалізації зведе тест нанівець
- Для демонстрації виведемо виклик події в окремий захищений віртуальний метод та викличемо його

```
        // Write the log to a storage
        // ...

        OnErrorLogged(Guid.NewGuid());
    }

    protected virtual void OnErrorLogged(Guid errorId)
    {
        ErrorLogged?.Invoke(this, errorId);
    }
```

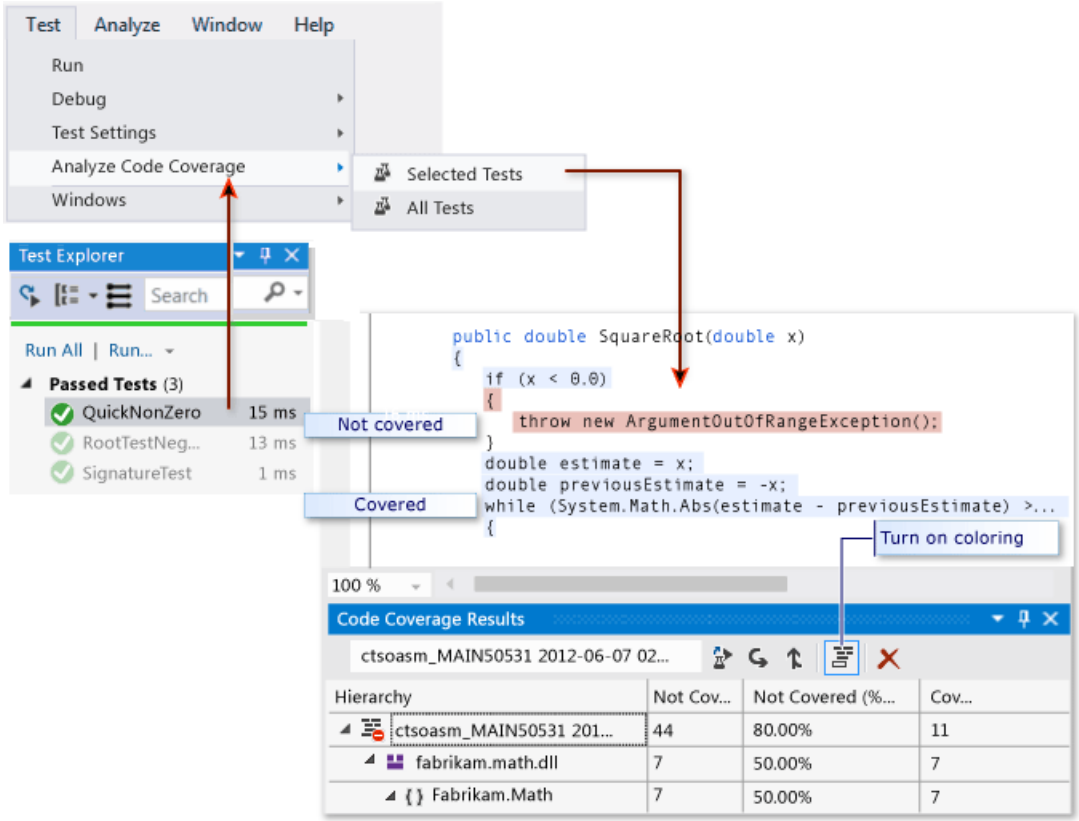
Тестувати потрібно метод Log(), а не OnErrorLogged()

Покриття коду тестами (Code Coverage)

Доступне у Visual Studio Enterprise Edition

- Частини коду можна виключати з перевірки покриття тестами:

```
public class ExampleClass1
{
    [ExcludeFromCodeCoverage]
    void ExampleMethod() {...}

    [ExcludeFromCodeCoverage] // exclude property
    int ExampleProperty1
    { get {...} set{...}}

    int ExampleProperty2
    {
        get
        {
            ...
        }
        [ExcludeFromCodeCoverage] // exclude setter
        set
        {
            ...
        }
    }

}
[ExcludeFromCodeCoverage]
class ExampleClass2 { ... }
```

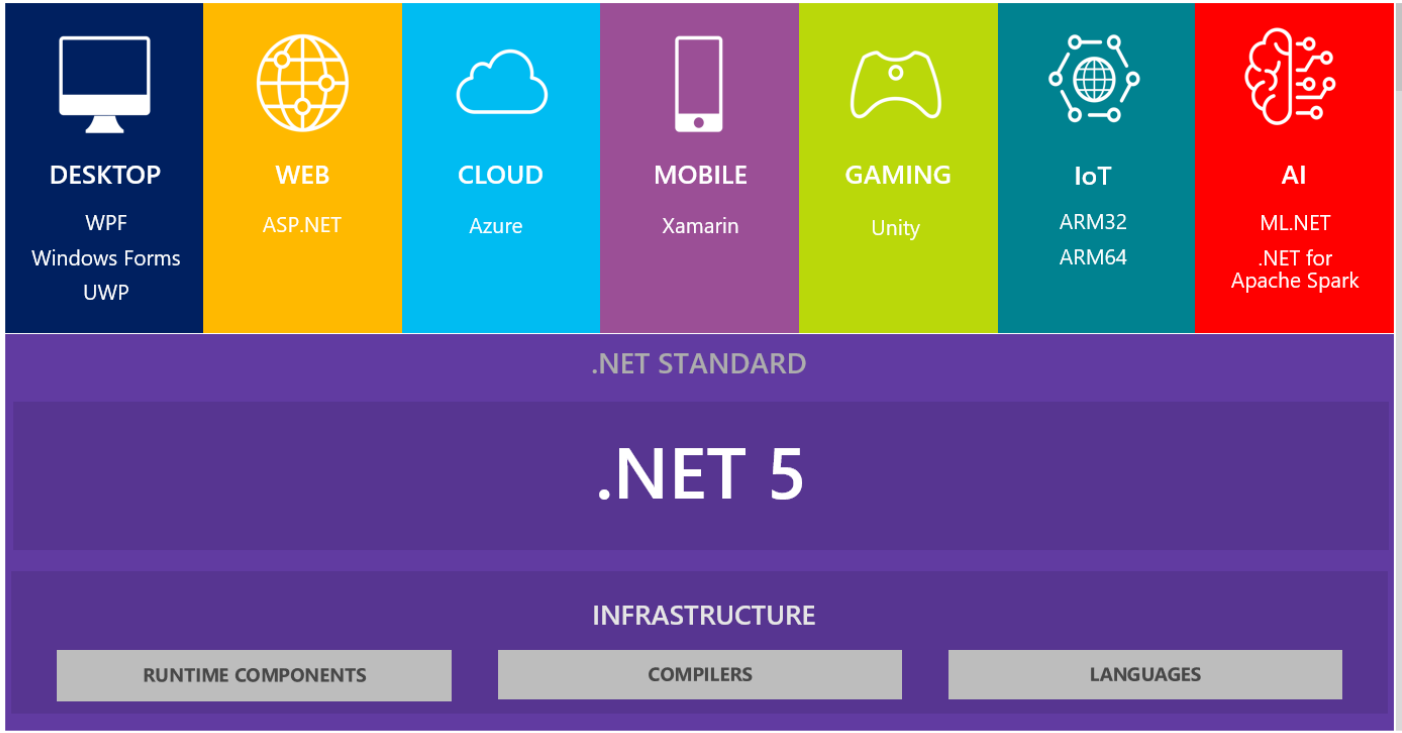


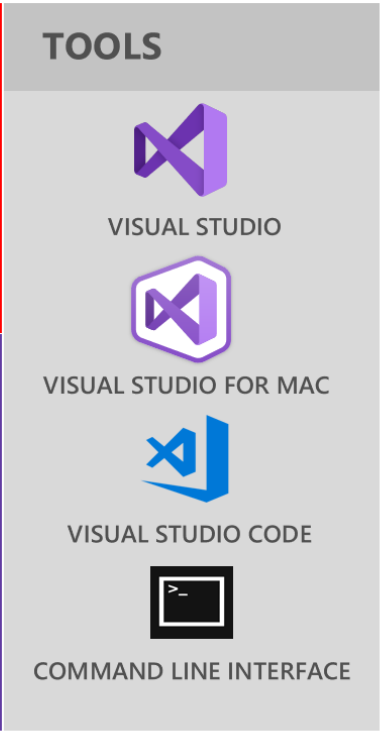
Самостійне опрацювання

- Приватні та захищені атрибути класу описують деталі реалізації.
  - Якщо написати тест до них, вони з ним стануть зв'язаними (coupling)
  - Зміна деталей реалізації зведе тест нанівець
- Для демонстрації виведемо виклик події в окремий захищений віртуальний метод та викличемо його

Уніфікована платформа .NET

Погляд у майбутнє





Плани щодо еволюції .NET

- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed